Relation Extraction with Weak Supervision and Distributional Semantics

by

Bonan Min

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
May 2013

Professor Ralph Grishman

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 2013**	2. REPORT TYPE	3. DATES COVERED **00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE

Relation Extraction with Weak Supervision and Distributional Semantics

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

New York University,Department of Computer Science,New York,NY,10012

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Relation Extraction aims at detecting and categorizing semantic relations be- tween pairs of entities in unstructured text. It benefits an enormous number of applications such as Web search and Question Answering. Traditional approaches for relation extraction either rely on learning from a large number of accurate human-labeled examples or pattern matching with hand-crafted rules. These re- sources are very laborious to obtain and can only be applied to a narrow set of target types of interest. This dissertation focuses on learning relations with little or no human supervi- sion. First, we examine the approach that treats relation extraction as a supervised learning problem. We develop an algorithm that is able to train a model with ap- proximately 1/3 of the human-annotation cost and that matches the performance of models trained with high-quality annotation. Second, we investigate distant supervision, a weakly supervised algorithm that automatically generates its own labeled training data. We develop a latent Bayesian framework for this purpose. By using a model which provides a better approximation of the weak source of supervision, it outperforms the state-of-the-art methods. Finally, we investigate the possibility of building all relational tables beforehand with an unsupervised relation extraction algorithm. We develope an effective yet efficient algorithm that combines the power of various semantic resources that are automatically mined from a corpus based on distributional semantics. The algorithm is able to extract a very large set of relations from the web at high precision.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**103**	

Dedicated to my parents.

Acknowledgements

First of all, I would like to thank my advisor Prof. Ralph Grishman. Ralph introduced me to the fascinating world of Information Extraction and taught me how to do research. I'm extremely grateful for his patience, especially for answering my questions in the late night, helping me with proofreading my draft papers even got it at the last minute. Ralph is always very supportive through my PhD study. This dissertation won't be possible without his guidance.

Second, I would like to thank my other mentors in the proteus group: Prof. Satoshi Sekine and Prof. Adam Meyers. Satoshi is an insipirasion to several of my research projects. Adam is the linguist that I always asked for help when I had questions. I would also like to thank my other comittee members: Prof. Heng Ji, Prof. Ernest Davis, and Prof. Dennis Shasha.

Third, I am indebt to two former proteus group members, Ang Sun and Shasha Liao. Their knowledge and passion on Information Extraction has directly led me to choose the field for my PhD. I'm especially grateful to Ang, who shared with me his code which is an excellent example of language engineering and the entry point to one of my research projects. I would like to thank following current members of the proteus group: Yifan He, Xiang Li, Maria Pershina, Lisheng Fu, Chen Chen, Wei Xu, Siyuan Zhou, Thien Huu Nguyen, Zachary Glass, Angus Grieve-Smith. I benefit tremendously from the discussion with them.

Fourth, I feel extremely lucky to have had two very fruitful summer internships. I would like to thank Dr. Chin-Yew Lin and Dr. Shuming Shi from Microsoft Research Asia (MSRA), and Dr. Chang Wang and Dr. David Gondek from IBM T.J. Watson Research Center. A large part of this dissertation is the result of collaboration with them. I would like to thank following researchers in MSRA: Dr.

Yunbo Cao, Dr. Wei Lai, Dr. Xian-Sheng Hua and Dr. Jun'ichi Tsujii. I would also like to thank everyone in the DeepQA team of IBM Research, including but not limited to: Dr. James Fan, Dr. Chris Welty, Dr. Alo Gliozzo, Dr. Aditya Kalyanpur, Dr. Sugato Bagchi, Dr. Siddharth Patwardhan, Dr. Eric Brown, Dr. Jennifer Chu-Carroll, Dr. Ken Barker, Dr. Edward A. Epstein, Dr. William Murdock. I enjoyed the many group discussions, weekly error analysis sessions and the great coffee in the lab.

Furthermore, I thank following (former) colleages at NYU: Prof. Jinyang Li, Prof. Lakshminarayanan Subramanian, Nguyen Tran, Yair Sovran, and Li Wan.

Finally I thank my parents for their love and support. I am most grateful for the support from my girlfriend Lan, without whom this dissertation would have been finished half a year earlier.

This work is supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058 and via Department of Interior National Business Center contract number D11PC20154. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, DoI/NBC, or the U.S. Government.

Abstract

Relation Extraction aims at detecting and categorizing semantic relations between pairs of entities in unstructured text. It benefits an enormous number of applications such as Web search and Question Answering. Traditional approaches for relation extraction either rely on learning from a large number of accurate human-labeled examples or pattern matching with hand-crafted rules. These resources are very laborious to obtain and can only be applied to a narrow set of target types of interest.

This dissertation focuses on learning relations with little or no human supervision. First, we examine the approach that treats relation extraction as a supervised learning problem. We develop an algorithm that is able to train a model with approximately 1/3 of the human-annotation cost and that matches the performance of models trained with high-quality annotation. Second, we investigate distant supervision, a weakly supervised algorithm that automatically generates its own labeled training data. We develop a latent Bayesian framework for this purpose. By using a model which provides a better approximation of the weak source of supervision, it outperforms the state-of-the-art methods. Finally, we investigate the possibility of building all relational tables beforehand with an unsupervised relation extraction algorithm. We develope an effective yet efficient algorithm that combines the power of various semantic resources that are automatically mined from a corpus based on distributional semantics. The algorithm is able to extract a very large set of relations from the web at high precision.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have dreamed of building a computer system that can understand human lanauge since the early days of modern computing. An area of Natural Language Processing (NLP) research that evolves towards this ultimate goal is Information Extraction (IE). IE involves developing algorithms that can automatically process unstructured text and generate a database of entities, relations and events. The resulting database allows computer algorithms to query and perform logical reasoning. IE not only unveils the intrinsic semantic meanings of text and moves us closer to the ultimate goal of enabling computers to understand text, but it can also be used for an enormous number of applications, such as web search and question answering.

One of the three core IE tasks is *Relation Extraction*, which aims at detecting and categorizing semantic relations between pairs of entities in text. The following are three examples of semantic relations:

- *(book_author)* ***Moby Dick*** *was written by* ***Hermann Melville.***
- *(located_in)* ***New York University*** *is located in* ***New York City****.*

- *(school_attended)* ***Bonan Min*** *studied at* ***New York University****.*

Popular approaches for relation extraction include 1) developing a machine learning algorithm that learns from a large amount of accurate human-annotated examples and can make predictions on new mentions, 2) using rules that are manually crafted or generated semi-automatically to match text fragments (raw or preprocessed). In addition, semantic knowledge resources (lexicons and paraphrase collections) have been shown to be important to the performance of such systems. All these human-generated sources of supervision (labeled examples, rules, and semantic resources) are very laborious to obtain and can only be applied to a narrow set of target types of interest.

This dissertation focuses on learning relations with little or no human supervision. First, we examine the approach that treats relation extraction as a supervised learning problem. We develop an algorithm that is able to train a model with low-cost annotation (1/3 of the normal cost) and that matches the performance of models trained with high-quality annotation. Second, we investigate distant supervision (DS), a weakly supervised algorithm that automatically generates its own labeled training data. We found that dealing with incorrectly labeled examples is critical for its success. We develop a latent Bayesian framework for this purpose. By using a model which provides a better approximation of the weak source of supervision, it outperforms the state-of-the-art methods. Finally, in collaboration with Microsoft Research Asia, we investigate the possibility of building all relational tables beforehand with an unsupervised relation extraction algorithm. We develope an effective yet efficient algorithm that combines the power of various semantic resources that are automatically mined from a corpus based on distributional semantics. The algorithm is able to extract a very large set of relations from the web at high precision.

The rest of the dissertation is organized as follows: first we review prior work in chapter 2, then we present three concrete solutions in chapter 3, 4 and 5, finally we conclude in chapter 6 and describe our immediate future work.

Chapter 2

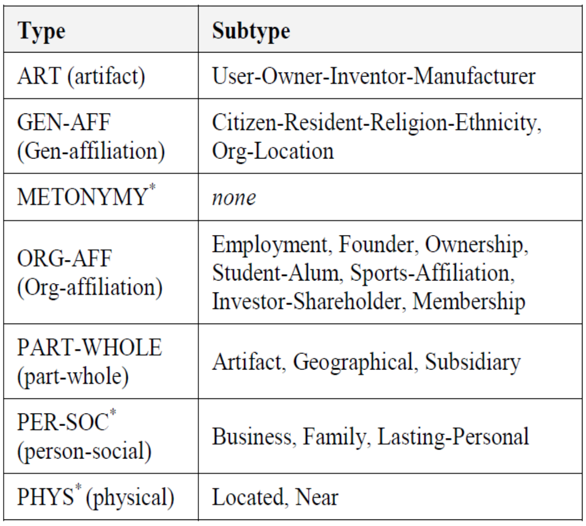
Prior Work

In this chapter, we review prior work that is closely related to the solutions (chapter 3,4,5) described in this dissertation. Our goal is to set the background for our work, but not to present a comprehensive survey of research in relation extraction. Therefore, certain interesting research in relation extraction, e.g., semi-supervised relation extraction [1] [6], is not described here. We refer interested readers to a recent survey [22] for further details on these methods.

2.1 Supervised relation extraction

Supervised methods ([38] [32] [7] [23] [83] [30] [60] [77] [9] [10] [82] [78] [79] [84] [50]) for relation extraction have been studied extensively since rich annotated linguistic resources were released. One of the most studied relation extraction tasks is the Automatic Content Extraction (ACE) relation extraction evaluation sponsored by the U.S. government. ACE 2005 defined 7 major entity types, such as PER (Person), LOC (Location), ORG (Organization), and also defines 7 major relation types and more than 20 subtypes (Figure 2.1[1]). ACE provides a large corpus which is manually annotated with entities (with coreference chains between entity mentions annotated), relations, events and values.

In this dissertation, we will call the text which is a mention of a entity as *entity mention*, following the ACE definition. There could be multiple mentions, taking the form of names/common nouns/pronouns, of an entity. Similarly, we also make a distinction between *relations* and *relation mentions*. A *relation mention* is a segment of text expressing a *relation* between two *entity mentions*. These two entity mentions are called the *arguments* of the relation mention. In ACE, each relation mention, expressing one of the predefined types, is tagged with a pair of entity mentions appearing in the same sentence as its arguments. More details about the ACE evaluation can be found on the official ACE website.[2]



Type	**Subtype**
ART (artifact)	User-Owner-Inventor-Manufacturer
GEN-AFF (Gen-affiliation)	Citizen-Resident-Religion-Ethnicity, Org-Location
METONYMY*	*none*
ORG-AFF (Org-affiliation)	Employment, Founder, Ownership, Student-Alum, Sports-Affiliation, Investor-Shareholder, Membership
PART-WHOLE (part-whole)	Artifact, Geographical, Subsidiary
PER-SOC* (person-social)	Business, Family, Lasting-Personal
PHYS* (physical)	Located, Near

Figure 2.1: ACE05 relation types and subtypes (Relations marked with an * are symmetric relations).

[1] The ACE 2005 (ACE05) Evaluation Plan: http://www.itl.nist.gov/iad/mig/tests/ace/ace05/doc/ace05-evalplan.v2a.pdf

[2] http://www.itl.nist.gov/iad/mig//tests/ace/ace05/

A supervised system for relation extraction has three steps: 1) data representation for labeled examples (a.k.a. relation mentions in ACE terms), e.g., feature extraction for feature-based method, or extracting objects for kernel-based method, 2) train a classification model as the relation detector/classifier, 3) apply the model as the relation extractor on the unseen relation mentions.

For data representation, state-of-the-art methods are either feature-based, or object-based. Given a relation mention, feature-based methods ([38] [32] [7] [23] [83] [30] [60]) extract a rich list of structural, lexical, syntactic and semantic features to represent it. [30] presents a systematic exploration of the relation feature space. In contrast, kernel based methods ([77] [9] [10] [82] [78] [79] [84] [50]) represent each instance with an object such as augmented token sequences or a parse tree, and use a carefully designed kernel function, e.g., subsequence kernel ([10]) or convolution tree kernel ([15]), to calculate their similarity. These objects are usually augmented with features such as semantic features.

Various machine learning methods have been applied for relation extraction. The two popular ones are maximum entropy classifiers (MaxEnt) ([32] [83] [30] [60]) and support vector machines (SVM) ([77] [30] [9] [10] [82] [78] [79] [84] [50]). Other methods such as K-nearest neighbors algorithm ([23]) and Voted Perceptron learning algorithm ([77]) have also been applied to the task [23].

2.2 Distant supervision for relation extraction

Distant supervision was first proposed by [12]. The approach generates weakly labeled examples by aligning facts in the Yeast Protein Database into the articles that may establish the facts, for training an extractor. Since then, it has gained popularity ([41] [11] [72] [51] [27] [59]). [11] treats each automatically-labeled relation mention as a labeled example, and trains an extractor with supervised learning that tolerates incorrect labels of positive examples. To provide a more accurate treatment of label noise while capturing the pair-level label constrains, [51] proposes to use Multiple Instance Learning ([16]), which assumes that only at-least-one of the mentions for each argument pair listed as having a relation in the KB, indeed has the target relation. MultiR ([27]) and Multi-Instance Multi-Label (MIML) learning ([59]) further improve it to allow a pair to have multiple relations. Their models allow different mentions for a pair to express different relations. [63] view the problem differently and propose a method that estimates the probability of each pattern showing each relation, based on the automatically labeled dataset. Their algorithm removes false positive matches by filtering mentions with low-probability patterns. [61] and [40] also estimate the probablities of patterns showing relations, but instead use them to relabel examples to their most likely classes. Their approach can correct highly-confident false negative matches.

Labeling noise can be reduced by using a more restricted labeling heuristic. For example, [71] assumes only the first sentence in Wikipedia text that contains a pair of related entities is a valid relation mention of the corresponding type. KYLIN [72] proposes three heuristics for labeling Wikipedia text with infobox. Such heuristics are able to improve label precision by sacrificing recall. Furthermore, they are domain-specific and are not applicable in a more distant yet common labeling scenario: align Freebase to newswires.

[71] uses distant supervision to improve supervised relation extraction. Their method starts with constructing relation topics from the set of heuristically labeled examples (by distant supervision) using Diffusion Wavelets. They propose a new

SVM kernel that encodes the background knowledge (a set of relation topics) as a source for measuring similarity between relation mentions. The resulting extraction algorithm, improves on existing solutions in the Automatic Content Extraction (ACE) relation evaluation dataset.

Notwithstanding this progress in distant supervision, current performance is still quite modest and not satisfactory for practical use. For example, the system very recently described in [59] achieves only a recall of 26.9 and a precision of 29.7 on a standard test set, Knowledge Base Population [29].

2.3 Unsupervised relation extraction

Unsupervised relation extraction (URE) algorithms ([25] [14] [55]) collect pairs of co-occurring entities as relation instances, extract features for instances and then apply unsupervised clustering techniques to find the major relations of a corpus. These UREs rely on tagging a predefined set of argument types, such as Person, Organization, and Location, in advance. [75] proposes several generative models, largely similar to LDA [5], for relation extraction. One of their models learns fine-grained semantic classes as relation arguments, but they share the similar requirement of tagging coarse-grained argument types. Most UREs use a quadratic clustering algorithm such as Hierarchical Agglomerate Clustering ([25] [55]), K-Means ([14]), or both ([52]) thus they are not scalable to very large corpora.

As the target domain shifts to the Web, new methods are proposed without requiring predefined entity types. Resolver ([76]) resolves objects and relation synonyms. [33] proposes Semantic Network Extractor (SNE) to extract concepts and relations. Based on second-order Markov logic, SNE uses a bottom-up agglomera-

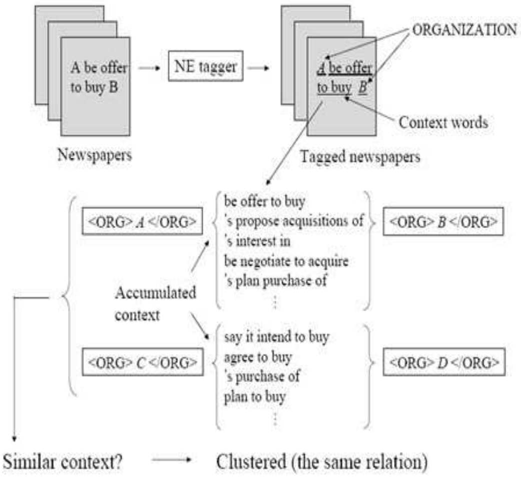


Figure 2.2: Unsupervised relation extraction[25].

tive clustering algorithm to jointly cluster relation phrases and argument entities. However, both Resolver and SNE require each entity and relation phrase to belong to exactly one cluster. This limits their ability to handle polysemous relation phrases. Moreover, SNE only uses features in the input set of relation instances for clustering, thus it fails to group many relevant instances. Resolver has the same sparseness problem but it is not affected as much as SNE because of its different goal (synonym resolution).

Chapter 3

Compensating for Annotation Errors in Training a Relation Extractor

The well-studied supervised relation extraction algorithms require training data that is accurate and has good coverage. To obtain such a gold standard, the common practice is to do independent double annotation followed by adjudication. This takes significantly more human effort than annotation done by a single annotator. We do a detailed analysis on a snapshot of the ACE 2005 annotation files to understand the differences between single-pass annotation and the more expensive nearly three-pass process, and then propose an algorithm that learns from the much cheaper single-pass annotation and achieves a performance on a par with the extractor trained on multi-pass annotated data. Furthermore, we show that given the same amount of human labor, the better way to do relation annotation is not to annotate with high-cost quality assurance, but to annotate more.

3.1 Introduction

Supervised methods for relation extraction have been studied extensively since rich annotated linguistic resources, e.g. the Automatic Content Extraction[1] (ACE) training corpus, were released. Those methods rely on accurate and complete annotation. To obtain high quality annotation, the common wisdom is to let two annotators independently annotate a corpus, and then asking a senior annotator to adjudicate the disagreements[2]. This annotation procedure roughly requires 3 passes[3] over the same corpus. Therefore it is very expensive. The ACE 2005 annotation on relations is conducted in this way.

In this chapter, we analyzed a snapshot of ACE training data and found that each annotator missed a significant fraction of relation mentions and annotated some spurious ones. We found that it is possible to separate most missing examples from the vast majority of true-negative unlabeled examples, and in contrast, most of the relation mentions that are adjudicated as incorrect contain useful expressions for learning a relation extractor. Based on this observation, we propose an algorithm that purifies negative examples and applies transductive inference to utilize missing examples during the training process on the single-pass annotation. Results show that the extractor trained on single-pass annotation with the proposed algorithm has a performance that is close to an extractor trained on the 3-pass annotation. We further show that the proposed algorithm trained on a single-pass annotation on the complete set of documents has a higher performance

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

[2] The senior annotator also found some missing examples as shown in Figure 3.1.

[3] In this chapter, we will assume that the adjudication pass has a similar cost compared to each of the two first-passes. The adjudicator may not have to look at as many sentences as an annotator, but he is required to review all instances found by both annotators. Moreover, he has to be more skilled and may have to spend more time on each instance to be able to resolve disagreements.

than an extractor trained on 3-pass annotation on 90% of the documents in the same corpus, although the effort of doing a single-pass annotation over the entire set costs less than half that of doing 3 passes over 90% of the documents. From the perspective of learning a high-performance relation extractor, it suggests that a better way to do relation annotation is not to annotate with a high-cost quality assurance, but to annotate more.

3.2 Background

3.2.1 Supervised Relation Extraction System

Given a sentence s and two entity mentions arg_1 and arg_2 contained in s, a candidate relation mention r with argument arg_1 preceding arg_2 is defined as $r = (s, arg_1, arg_2)$. The goal of Relation Detection and Classification (RDC) is to determine whether r expresses one of the types defined. If so, classify it into one of the types. Supervised learning treats RDC as a classification problem and solves it with supervised Machine Learning algorithms such as MaxEnt and SVM. There are two commonly used learning strategies ([60]). Given an annotated corpus, one could apply a **flat** learning strategy, which trains a single multi-class classifier on training examples labeled as one of the relation types or *not-a-relation*, and apply it to determine its type or output *not-a-relation* for each candidate relation mention during testing. The examples of each type are the relation mentions that are tagged as instances of that type, and the *not-a-relation* examples are constructed from pairs of entities that appear in the same sentence but are not tagged as any of the types. Alternatively, one could apply a **hierarchical** learning strategy, which trains two classifiers, a binary classifier RD for relation detection and the other a multi-class classifier RC for relation classification. RD is trained by grouping tagged relation mentions of all types as positive instances and using all the not-a-relation cases (same as described above) as negative examples. RC is trained on the annotated examples with their tagged types. During testing, RD is applied first to identify whether an example expresses some relation, then RC is applied to determine the most likely type only if it is detected as correct by RD.

Following previous work, we ignore sub-types in this chapter and only evaluate on types when reporting relation classification performance. We use the hierarchical learning strategy since it simplifies the problem by letting us focus on relation detection only. The relation classification stage remains unchanged and we will show that it benefits from improved detection. For experiments on both relation detection and relation classification, we use SVM[4] [65] as the learning algorithm since it can be extended to support transductive inference as discussed in chapter 3.4.3. However, for the analysis in chapter 3.3.2 and the purification preprocess steps in chapter 3.4.2, we use a MaxEnt[5] model since it outputs probabilities[6] for its predictions. For the choice of features, we use the full set of features from [83] since it is reported to have a state-of-the-art performance [60]. Finally we use the perfect entity mentions instead of mentions detected by a Named Entity Recognition algorithm, following the common practice (e.g., [83]).

[4] SVM-Light is used. http://svmlight.joachims.org/

[5] OpenNLP MaxEnt package is used. http://maxent.sourceforge.net/about.html

[6] SVM also outputs a value associated with each prediction. However, this value cannot be interpreted as probability.

3.2.2 ACE 2005 annotation

The ACE 2005 training data contains 599 articles from newswire, broadcast news, weblogs, usenet newsgroups/discussion forum, conversational telephone speech and broadcast conversations. The annotation process is conducted as follows: two annotators working independently annotate each article and complete all annotation tasks (entities, values, relations and events). After two annotators both finished annotating a file, all discrepancies are then adjudicated by a senior annotator. This results in a high-quality annotation file. More details can be found in the documentation of ACE 2005 Multilingual Training Data V3.0[7]. Since the final release of the ACE training corpus only contains the final adjudicated annotations, in which all the traces of the two first-pass annotations are removed, we use a snapshot of almost-finished annotation, ACE 2005 Multilingual Training Data V3.0, for our analysis. In the remainder of this chapter, we will call the two independent first-passes of annotation *fp1* and *fp2*. The higher-quality data done by merging *fp1* and *fp2* and then having disagreements adjudicated by the senior annotator is called *adj*. From this corpus, we removed the files that have not been completed for all three passes. On the final corpus consisting of 511 files, we can differentiate the annotations on which the three annotators have agreed and disagreed. A notable fact of ACE relation annotation is that it is done with arguments from the list of annotated entity mentions. For example, in a relation mention *tyco's ceo and president dennis kozlowski* which expresses an *EMP-ORG* relation, the two arguments *tyco* and *dennis kozlowski* must have been tagged as entity mentions previously by the annotator. Since *fp1* and *fp2* are done on all tasks independently, their disagreement on entity annotation will be propagated

[7] LDC2005E18. LDC Catalog

to relation annotation; thus we need to deal with these cases specifically.

3.3 Analysis of data annotation

3.3.1 General statistics

As discussed in chapter 3.2, relation mentions are annotated with entity mentions as arguments, and the lists of annotated entity mentions vary in *fp1*, *fp2* and *adj*. To estimate the impact propagated from entity annotation, we first calculate the ratio of overlapping entity mentions between entities annotated in *fp1*/*fp2* with *adj*. We found that *fp1*/*fp2* each agrees with *adj* on around 89% of the entity mentions. Following up, we checked the relation mentions[8] from *fp1* and *fp2* against the adjudicated list of entity mentions from adj and found that 682 and 665 relation mentions respectively have at least one argument which doesn't appear in the list of adjudicated entity mentions.

Given the list of relation mentions with both arguments appearing in the list of adjudicated entity mentions, Figure 3.1 shows the inter-annotator agreement of the ACE 2005 relation annotation. In this figure, the three circles represent the list of relation mentions in *fp1*, *fp2* and *adj*, respectively.

It shows that each annotator missed a significant number of relation mentions annotated by the other. Considering that we removed 682/665 relation mentions from *fp1*/*fp2* because we generate this figure based on the list of adjudicated entity mentions, we estimate that *fp1* and *fp2* both missed around 18.3-28.5%[9] of

[8] This is done by selecting the relation mentions whose both arguments are in the list of adjudicated entity mentions.

[9] We calculate the lower bound by assuming that the 682 relation mentions removed from *fp1* are found in *fp2*, although with different argument boundary and headword tagged. The upper bound is calculated by assuming that they are all irrelevant and erroneous relation mentions.

the relation mentions. This clearly shows that both of the annotators missed a significant fraction of the relation mentions. They also annotated some spurious relation mentions (as adjudicated in *adj*), although the fraction is smaller (close to 10% of all relation mentions in *adj*).

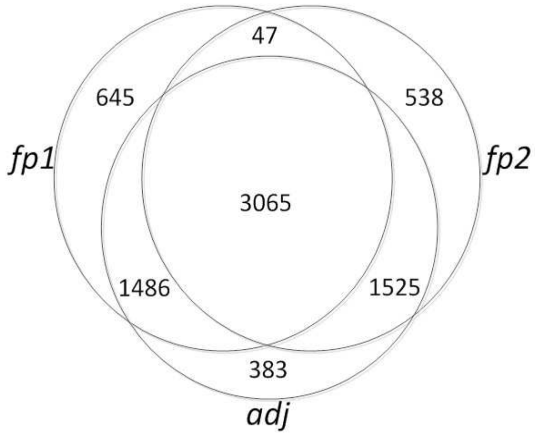


Figure 3.1: Inter-annotator agreement of ACE 2005 relation annotation. Numbers are the distinct relation mentions whose both arguments are in the list of adjudicated entity mentions.

ACE 2005 relation annotation guidelines (ACE English Annotation Guidelines for Relations, version 5.8.3[10]) defined 7 syntactic classes and the *other* class. We plot the distribution of syntactic classes of the annotated relations in Figure 3.2 (3 of the classes, accounting together for less than 10% of the cases, are omitted) and the *other* class. It shows that it is generally easier for the annotators to find and agree on relation mentions of the type *Preposition/PreMod/Possessives* but harder to find and agree on the ones belonging to *Verbal* and *Other*. The definition and examples of these syntactic classes can be found in the annotation guidelines.

In the following sections, we will show the analysis on *fp1* and *adj* since the result is similar for *fp2*.

[10] http://projects.ldc.upenn.edu/ace/

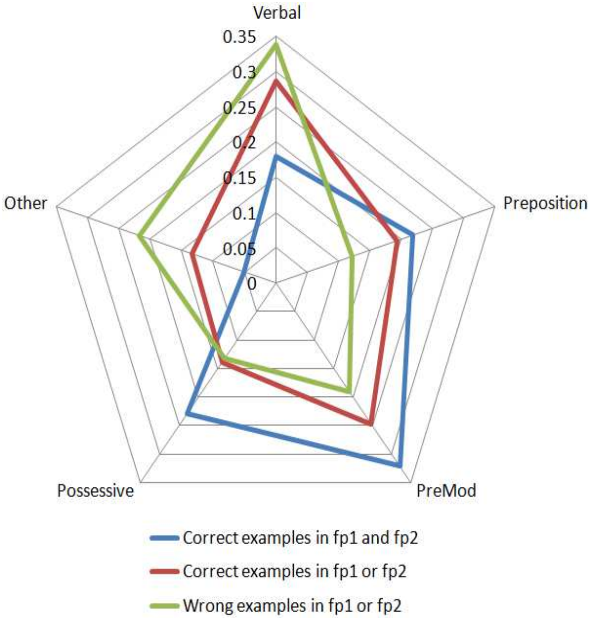


Figure 3.2: Percentage of examples of major syntactic classes.

3.3.2 Why the differences?

To understand what causes the missing annotations and the spurious ones, we need methods to find how similar/different the false positives are to true positives and also how similar/different the false negatives (missing annotations) are to true negatives. If we adopt a good similarity metric, which captures the structural, lexical and semantic similarity between relation mentions, this analysis will help us to understand the similarity/difference from an extraction perspective.

We use a state-of-the-art feature space [83] to represent examples (including all correct examples, erroneous ones and untagged examples) and use MaxEnt as the weight learning model since it shows competitive performance in relation extraction [30] and outputs probabilities associated with each prediction. We train a MaxEnt model for relation detection on true positives and true negatives, which respectively are the subset of correct examples annotated by *fp1* (and adjudicated as correct ones) and negative examples that are not annotated in *adj*, and use it to make predictions on the mixed pool of correct examples, missing examples and spurious ones.

To illustrate how distinguishable the missing examples (false negatives) are from the true negative ones, 1) we apply the MaxEnt model on both false negatives and true negatives, 2) put them together and rank them by the model-predicted probabilities of being positive, 3) calculate their relative rank in this pool. We plot the Cumulative distribution of frequency (CDF) of the ranks (as percentages in the mixed pools) of false negatives in figure 3. We took similar steps for the spurious ones (false positives) and plot them in figure 3 as well (However, they are ranked by model-predicted probabilities of being negative).

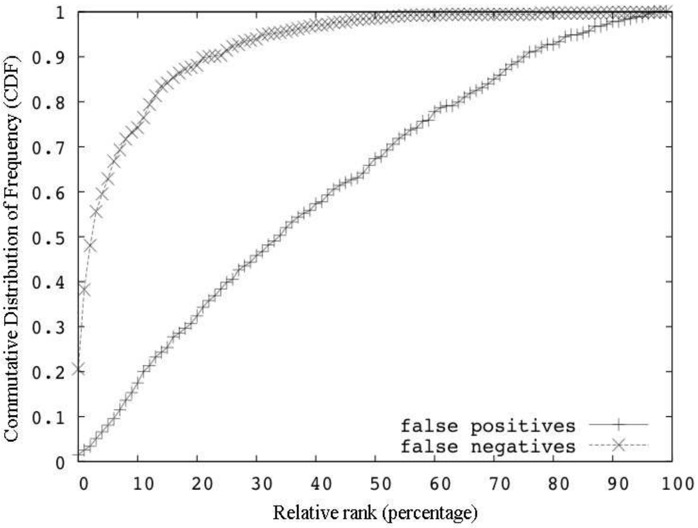


Figure 3.3: Cumulative distribution of frequency (CDF) of the relative ranking of model-predicted probability of being positive for false negatives in a pool mixed of false negatives and true negatives; and the CDF of the relative ranking of model-predicted probability of being negative for false positives in a pool mixed of false positives and true positives.

For false negatives, it shows a highly skewed distribution in which around 75% of the false negatives are ranked within the top 10%. That means the missing examples are lexically, structurally or semantically similar to correct examples, and are distinguishable from the true negative examples. However, the distribution of false positives (spurious examples) is close to uniform (flat curve), which means they are generally indistinguishable from the correct examples.

3.3.3 Categorize annotation errors

The automatic method shows that the errors (spurious annotations) are very similar to the correct examples but provides little clue as to why that is the case. To understand their causes, we sampled 65 examples from *fp1* (10% of the 645 errors), read the sentences containing these erroneous relation mentions and compared them to the correct relation mentions in the same sentence; we categorized these examples and show them in Table 3.1. The most common type of error is *duplicate relation mention for coreferential entity mentions.* The first row in Table 3.1 shows an example, in which there is a relation ORG-AFF tagged between *US* and *George W. Bush* in *adj.* Because *President* and *George W. Bush* are coreferential, the example <*US, President*> from *fp1* is adjudicated as incorrect. This shows that if a relation is expressed repeatedly across relation mentions whose arguments are coreferential, the adjudicator only tags one of the relation mentions as correct, although the other is correct too. This shared the same principle with another type of error *illegal promotion through blocked categories*[11] as defined in the annotation

[11] For example, in sentence *Smith went to a hotel in Brazil*, *(Smith, hotel)* is a taggable PHYS Relation but *(Smith, Brazil)* is not, because to get the second relationship, one would have to promote *Brazil* through *hotel.* For the precise definition of annotation rules, please refer to ACE (Automatic Content Extraction) English Annotation Guidelines for Relations, version 5.8.3.

guideline. The second largest category is *correct*, by which we mean the example is a correct relation mention and the adjudicator made a mistake. The third largest category is *argument not in list*, by which we mean that at least one of the arguments is not in the list of adjudicated entity mentions.

Based on Table 3.1, we can see that as many as 72%-88% of the examples which are adjudicated as incorrect are actually correct if viewed from a relation learning perspective, since most of them contain informative expressions for tagging relations. The annotation guideline is designed to ensure high quality while not imposing too much burden on human annotators. To reduce annotation effort, it defined rules such as *illegal promotion through blocked categories*. The annotators practice suggests that they are following another rule not to annotate *duplicate relation mention for coreferential entity mentions*. This follows the similar principle of reducing annotation effort but is not explicitly stated in the guideline: to avoid propagation of a relation through a coreference chain. However, these examples are useful for learning more ways to express a relation. Moreover, even for the erroneous examples (as shown in Table 3.1 as *violate reasonable reader rule and errors*), most of them have some level of similar structures or semantics to the targeted relation. Therefore, it is very hard to distinguish them without human proofreading.

3.3.4 Why missing annotations and how many examples are missing?

For the large number of missing annotations, there are a couple of possible reasons. One reason is that it is generally easier for a human annotator to annotate correctly given a well-defined guideline, but it is hard to ensure completeness, es-

Category	Percentage	Example		
		Relation Type	Sampled text of spurious examples in *fp1*	Notes (examples are similar ones in *adj* for comparison)
Duplicate relation mention for coreferential entity mentions	49.2%	ORG-AFF	*his budding friendship with US President George W. Bush in the face of*	*his budding friendship with US President George W. Bush in the face of*
Correct	20%	PHYS	*Hundreds of thousands of demonstrators took to the streets in Britain*	
		PER-SOC	*The dead included the quack doctor, 55-year-old Nityalila Naotia, his teenaged son and*	(Symmetric relation) *The dead included the quack doctor, 55-year-old Nityalila Naotia, his teenaged son*
Argument not in list	15.4%	PER-SOC	*Putin had even secretly invited British Prime Minister Tony Blair, Bush's staunchest backer in the war on Iraq*	
Violate reasonable reader rule	6.2%	PHYS	*”The amazing thing is they are going to turn San Francisco into ground zero for every criminal who wants to profit at their chosen profession”, Paredes said.*	
Errors	6.1%	PART-WHOLE	*a likely candidate to run Vivendi Universal's entertainment unit in the United States*	Arguments are tagged reversed
		PART-WHOLE	*Khakamada argued that the United States would also need Russia's help ”to make the new Iraqi government seem legitimate.*	Relation type error
illegal promotion through blocked categories	3%	PHYS	*Up to 20,000 protesters thronged the plazas and streets of San Francisco, where*	*Up to 20,000 protesters thronged the plazas and streets of San Francisco, where*

Table 3.1: Categories of spurious relation mentions in fp1 (on a sample of 10% of relation mentions), ranked by the percentage of the examples in each category. In the sample text, dotted underlined text shows head words of the first arguments and the underlined text shows head words of the second arguments.

pecially for a task like relation extraction. Furthermore, the ACE 2005 annotation guideline defines more than 20 relation subtypes. These many subtypes make it hard for an annotator to keep all of them in mind while doing the annotation, and thus it is inevitable that some examples are missed.

Here we proceed to approximate the number of missing examples given limited knowledge. Let each annotator annotate n examples and assume that each pair of annotators agrees on a certain fraction p of the examples. Assuming the examples are equally likely to be found by an annotator, therefore the total number of unique examples found by k annotators is $\sum_{i=0}^{k} (1-p)^i n$. If we had an infinite number of annotators ($k \to \infty$), the total number of unique examples will be n/p , which is the upper bound of the total number of examples. In the case of the ACE 2005 relation mention annotation, since the two annotators annotate around 4500 examples and they agree on 2/3 of them, the total number of all positive examples is around 6750. This is close to the number of relation mentions in the adjudicated list: 6459. Here we assume the adjudicator is doing a more complex task than an annotator, resolving the disagreements and completing the annotation (as shown in figure 1).

The assumption of the calculation is a little crude but reasonable given the limited number of passes of annotation we have. Recent research [28] shows that, by adding annotators for IE tasks, the merged annotation tends to converge after having 5 annotators. To understand the annotation behavior better, in particular whether annotation will converge after adding a few annotators, more passes of annotation need to be collected. We leave this as future work.

3.4 Relation extraction with low-cost annotation

3.4.1 Baseline algorithm

To see whether a single-pass annotation is useful for relation detection and classification, we did 5-fold cross validation (5-fold CV) with each of *fp1*, *fp2* and *adj* as the training set, and tested on *adj*. The experiments are done with the same 511 documents we used for the analysis. As shown in Table 3.2, we did 5-fold CV on *adj* for experiment 3. For fairness, we use settings similar to 5-fold CV for experiment 1 and 2. Take experiment 1 as an example: we split both of *fp1* and *adj* into 5 folds, use 4 folds from *fp1* as training data, and 1 fold from *adj* as testing data and does one train-test cycle. We rotate the folds (both training and testing) and repeat 5 times. The final results are averaged over the 5 runs. Experiment 2 was conducted similarly. In the reminder of the chapter, 5-fold CV experiments are all conducted in this way.

Table 3.2 shows that a relation tagger trained on the single-pass annotated data *fp1* performs worse than the one trained on merged and adjudicated data *adj*, with 4.6 points lower F measure in relation detection, and 4.6 points lower relation classification. For detection, precision on *fp1* is 3 points higher than on *adj* but recall is much lower (close to 10 points). The recall difference shows that the missing annotations contain expressions that can help to find more correct examples during testing. The small precision difference indirectly shows that the spurious ones in *fp1* (as adjudicated) do not hurt precision. Performance on classification shows a similar trend because the relation classifier takes the examples predicted by the detector as correct as its input. Therefore, if there is an error, it gets propagated to this stage. Table 3.2 also shows similar performance differences between *fp2* and *adj*.

Exp #	Training	Testing	Detection (%)			Classification (%)		
			Precision	Recall	F1	Precision	Recall	F1
1	*fp1*	*adj*	83.4	60.4	70.0	75.7	54.8	63.6
2	*fp2*	*adj*	83.5	60.5	70.2	76.0	55.1	63.9
3	*adj*	*adj*	80.4	69.7	74.6	73.4	63.6	68.2

Table 3.2: Performance of RDC trained on *fp1/fp2/adj*, and tested on *adj*.

In the remainder of this chapter, we will discuss a few algorithms to improve a relation tagger trained on single-pass annotated data[12]. Since we already showed that most of the spurious annotations are not actually errors from an extraction perspective and table 2 shows that they do not hurt precision, we will only focus on utilizing the missing examples, in other words, training with an incomplete annotation.

3.4.2 Purify the set of negative examples

Traditional supervised methods find all pairs of entity mentions that appear within a sentence, and then use the pairs that are not annotated as relation mentions as the negative examples for the purpose of training a relation detector. It relies on the assumption that the annotators annotated all relation mentions and missed no (or very few) examples. However, this is not true for training on a single-pass annotation, in which a significant portion of relation mentions are left not annotated. If this scheme is applied, all of the correct pairs which the annotators missed belong to this negative category. Therefore, we need a way to purify the negative set of examples obtained by this conventional approach.

[12] We only use fp1 and adj in the following experiments because we observed that fp1 and fp2 are similar in general in the analysis, though a fraction of the annotation in fp1 and fp2 is different. Moreover, algorithms trained on them show similar performance.

[35] focuses on classifying documents with only positive examples. Their algorithm initially sets all unlabeled data to be negative and trains a Rocchio classifier, selects negative examples which are closer to the negative centroid than positive centroid as the purified negative examples, and then retrains the model. Their algorithm performs well for text classification. It is based on the assumption that there are fewer unannotated positive examples than negative ones in the unlabeled set, so true negative examples still dominate the set of noisy negative examples in the purification step.

Based on the same assumption, our purification process consists of the following steps:

1. Use annotated relation mentions as positive examples; construct all possible relation mentions that are not annotated, and initially set them to be negative. We call this noisy data set D.

2. Train a MaxEnt relation detection model M_{det} on D.

3. Apply M_{det} on all unannotated examples, and rank them by the model-predicted probabilities of being positive.

4. Remove the top N examples from D.

These preprocessing steps result in a purified data set D_{pure}. We can use D_{pure} for the normal training process of a supervised relation extraction algorithm.

The algorithm is similar to [35]. However, we drop a few noisy examples instead of choosing a small purified subset since we have relatively few false negatives compared to the entire set of unannotated examples. Moreover, after step 3, most false negatives are clustered within the small region of top ranked examples which has a high model-predicated probability of being positive. The intuition is similar to what we observed from Figure 3.3 for false negatives since we also observed very similar distribution using the model trained with noisy data. Therefore, we can purify negatives by removing examples in this noisy subset.

However, the false negatives are still mixed with true negatives. For example, still slightly more than half of the top 2000 examples are true negatives. Thus we cannot simply flip their labels and use them as positive examples. In the following section, we will use them in the form of unlabeled examples to help train a better model.

3.4.3 Transductive inference on unlabeled examples

Transductive SVM ([65] [31]) is a semi-supervised learning method which learns a model from a data set consisting of both labeled and unlabeled examples. Compared to its popular antecedent SVM, it also learns a maximum margin classification hyperplane, but additionally forces it to separate a set of unlabeled data with large margin. The optimization function of Transductive SVM (TSVM) is the following:

$$\begin{array}{ll}
\textit{Minimize over } (y_1^*, \ldots, y_n^*, \vec{w}, b, \xi_1, \ldots, \xi_n, \xi_1^*, \ldots, \xi_k^*): & \\
 & \frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^* \\
\textit{subject to:} & \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \\
 & \forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j^* + b] \geq 1 - \xi_j^* \\
 & \forall_{i=1}^{n} : \xi_i > 0 \\
 & \forall_{j=1}^{k} : \xi_j^* > 0
\end{array}$$

Figure 3.4: TSVM optimization function for non-separable case [31].

TSVM can leverage an unlabeled set of examples to improve supervised learning. As shown in chapter 3.3, a significant number of relation mentions are missing from the single-pass annotation data. Although it is not possible to find all missing annotations without human effort, we can improve the model by further utilizing the fact that some unannotated examples should have been annotated.

The purification process discussed in the previous section removes N examples which have a high density of false negatives. We further utilize the N examples as follows:

1. Construct a training corpus D_{hybrid} from D_{pure} by taking a random sample[13] of $N(1-p)/p$ (p is the ratio of annotated examples to all examples; $p = 0.05$ in $fp1$) negatively labeled examples in D_{pure} and setting them to be unlabeled. In addition, the N examples removed by the purification process are added back as unlabeled examples.

2. Train TSVM on D_{hybrid}.

The second step trained a model which replaced the detection model in the hierarchical detection-classification learning scheme we used. We will show in the next section that this improves the model.

3.5 Experiments

Experiments were conducted over the same set of documents on which we did analysis: the 511 documents which have completed annotation in all of the *fp1*, *fp2*

[13] We included this large random sample so that the balance of positive to negative examples in the unlabeled set would be similar to that of the labeled data. The test data is not included in the unlabeled set.

and *adj* from the ACE 2005 Multilingual Training Data V3.0. To reemphasize, we apply the hierarchical learning scheme and we focus on improving relation detection while keeping relation classification unchanged (results show that its performance is improved because of the improved detection). We use SVM as our learning algorithm with the full feature set from [83].

Baseline algorithm: The relation detector is unchanged. We follow the common practice, which is to use annotated examples as positive ones and all possible untagged relation mentions as negative ones. We sub-sampled the negative data by since that shows better performance.

+purify: This algorithm adds an additional purification preprocessing step (chapter 3.4.2) before the hierarchical learning RDC algorithm. After purification, the RDC algorithm is trained on the positive examples and purified negative examples. We set $N = 2000$[14] in all experiments.

+tSVM: First, the same purification process of **+purify** is applied. Then we follow the steps described in chapter 3.4.3 to construct the set of unlabeled examples, and set all the rest of purified negative examples to be negative. Finally, we train TSVM on both labeled and unlabeled data and replace the relation detection in the RDC algorithm. The relation classification is unchanged.

Table 3.3 shows the results. All experiments are done with 5-fold cross validation[15] using testing data from *adj*. The first three rows show experiments trained on *fp1*, and the last row (**ADJ**) shows the unmodified RDC algorithm trained on *adj* for comparison. The purification of negative examples shows significant perfor-

[14] We choose 2000 because it is close to the number of relations missed from each single-pass annotation. In practice, it contains more than 70% of the false negatives, and it is less than 10% of the unannotated examples. To estimate how many examples are missing (chapter 3.3.4), one should perform multiple passes of independent annotation on a small dataset and measure inter-annotator agreements.

[15] Details about the settings for 5-fold cross validation are in chapter 3.4.1.

mance gain, 3.7% F1 on relation detection and 3.4% on relation classification. The precision decreases but recall increases substantially since the missing examples are not treated as negatives. Experiment shows that the purification process removes more than 60% of the false negatives. Transductive SVM further improved performance by a relatively small margin. This shows that the latent positive examples can help refine the model. Results also show that transductive inference can find around 17% of missing relation mentions. We notice that the performance of relation classification is improved since by improving relation detection, some examples that do not express a relation are removed. The classification performance on single-pass annotation is close to the one trained on *adj* due to the help from a better relation detector trained with our algorithm.

We also did 5-fold cross validation with a model trained on a fraction of the 4/5 (4 folds) of *adj* data (each experiment shown in table 4 uses 4 folds of *adj* documents for training since one fold is left for cross validation). The documents are sampled randomly. Table 3.4 shows results for varying training data size. Compared to the results shown in the +tSVM row of Table 3.3, we can see that our best model trained on single-pass annotation outperforms SVM trained on 90% of the dual-pass, adjudicated data in both relation detection and classification, although it costs less than half the 3-pass annotation. This suggests that given the same amount of human effort for relation annotation, annotating more documents with single-pass offers advantages over annotating less data with high quality assurance (dual passes and adjudication).

Algorithm	Detection (%)			Classification (%)		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	83.4	60.4	70.0	75.7	54.8	63.6
+purify	76.8	70.9	73.7	69.8	64.5	67.0
+tSVM	76.4	72.1	74.2	69.4	65.2	67.2
ADJ (on *adj*)	80.4	69.7	74.6	73.4	63.6	68.2

Table 3.3: 5-fold cross-validation results. All are trained on *fp1* (except the last row showing the unchanged algorithm trained on *adj* for comparison), and tested on *adj*. McNemar's test show that the improvement from +purify to +tSVM, and from +tSVM to ADJ are statistically significant (with $p < 0.05$).

adj used	Detection (%)			Classification (%)		
	Precision	Recall	F1	Precision	Recall	F1
60% 4/5	86.9	41.2	55.8	78.6	37.2	50.5
70% 4/5	85.5	51.3	64.1	77.7	46.6	58.2
80% 4/5	83.3	58.1	68.4	75.8	52.9	62.3
90% 4/5	82.0	64.9	72.5	74.9	59.4	66.2

Table 3.4: Performance with SVM trained on a fraction of *adj*. It shows 5 fold cross validation results.

3.6 Related work

[17] studies WSD annotation from a cost-effectiveness viewpoint. They showed empirically that, with same amount of annotation dollars spent, single-annotation is better than dual-annotation and adjudication. The common practice for quality control of WSD annotation is similar to Relation annotation. However, the task of WSD annotation is very different from relation annotation. WSD requires that every example must be assigned some tag, whereas that is not required for relation tagging. Moreover, relation tagging requires identifying two arguments and correctly categorizing their types.

The purified approach applied in this chapter is related to the general framework of learning from positive and unlabeled examples. [35] initially set all unlabeled data to be negative and train a Rocchio classifier, then select negative examples which are closer to the negative centroid than positive centroid as the purified negative examples. We share a similar assumption with [35] but we use a different method to select negative examples since the false negative examples show a very skewed distribution, as described in chapter 3.3.2.

Transductive SVM was introduced by [65] and later refined in [31]. A few related methods were studied on the subtask of relation classification (the second stage of the hierarchical learning scheme) in [80].

[13] observed the similar phenomenon that ACE annotators rarely duplicate a relation link for coreferential mentions. They use an evaluation scheme to avoid being penalized by the relation mentions which are not annotated because of this behavior.

3.7 Conclusion

We analyzed a snapshot of the ACE 2005 relation annotation and found that each single-pass annotation missed around 18-28% of relation mentions and contains around 10% spurious mentions. A detailed analysis showed that it is possible to find some of the false negatives, and that most spurious cases are actually correct examples from a system builders perspective. By automatically purifying negative examples and applying transductive inference on suspicious examples, we can train a relation classifier whose performance is comparable to a classifier trained on the dual-annotated and adjudicated data. Furthermore, we show that single-pass annotation is more cost-effective than annotation with high quality assurance.

Chapter 4

Weakly Supervised Relation Extraction

Distant supervision, heuristically labeling a corpus using a knowledge base, has emerged as a popular choice for training relation extractors. However, it generates noisy class labels. In this chapter, we first analyze two problems of distant supervision, and then propose a general statistical framework which models both mention-level and entity-level noise. Experimental results demonstrate its advantage over existing algorithms.

4.1 Introduction

Recently, Distant Supervision (DS) [12] [41] has emerged to be a popular choice for training relation extractors without using manually labeled data. It automatically generates training examples by labeling relation mentions in the source corpus according to whether the argument pair is listed in the target relational tables in a knowledge base (KB). This method significantly reduces human efforts for relation extraction.

However, serious problems remain which limit the broad application of these weakly supervised methods. First, the labeling heuristics mapping a KB to unlabeled corpora have generated lots of incorrect labels. For example, mapping Freebase[1] to New York Times generates about 31% errors. Second, it also generates lots of false negatives because knowledge bases are highly incomplete (as high as 90% incompleteness for popular types such as the place of birth of a person in Freebase).

In this chapter, we analyze these two problems and present a novel statistical framework. Our weakly supervised learning framework jointly models mention-level and entity-level noise, and is able to address both problems. Experimental results on a realistic and challenging dataset demonstrate the advantage of the algorithm over existing solutions.

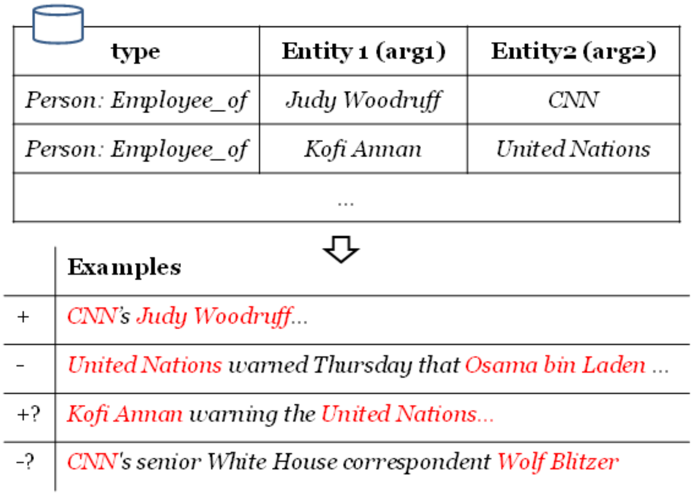
4.2 Problem definition

Distant Supervision: Given a KB $\mathcal{D}$ (a collection of relational tables $r(e_1, e_2)$, in which $r \epsilon R$ (R is the set of relation labels), and $< e_1, e_2 >$ is a pair of entities that is known to have relation r) and a corpus C, the key idea of distant supervision is that we align $\mathcal{D}$ to C, label each bag[2] of relation mentions that share argument pair $< e_1, e_2 >$ with r, otherwise OTHER. This generates a dataset that has labels on entity-pair (bag) level. Figure 4.1 illustrates the labeling process. After

[1] Freebase is a large collaboratively-edited KB. It is available at http://www.freebase.com.

[2] A bag is defined as a set of relation mentions sharing the same entity pair as relation arguments. We will use the terms bag and entity pair interchangeably in this chapter.

generating labeled examples, a relation extractor is trained with algorithms such as conventional supervised single-instance learning (by assuming all mentions have the same label of the bag), or Multiple-Instance Learning (by assuming at-least-one of the mentions expresses the bag-level label), or Multi-Instance Multi-Label learning (further assuming a bag can have multiple labels). All previous research treats the OTHER class as the negative class for training purpose.



type	Entity 1 (arg1)	Entity2 (arg2)
Person: Employee_of	*Judy Woodruff*	*CNN*
Person: Employee_of	*Kofi Annan*	*United Nations*
...		

	Examples
+	*CNN's Judy Woodruff...*
-	*United Nations warned Thursday that Osama bin Laden ...*
+?	*Kofi Annan warning the United Nations...*
-?	*CNN's senior White House correspondent Wolf Blitzer*

Figure 4.1: Generate labeled examples (bottom) with a KB (top) using distant supervision. + and - mark examples labeled as *positive* and *negative*, respectively, for the relation *Person:Employee_of*. +? and -? respectively mark the ones labeled positive and negative by the process, but in fact one can tell that they do not belong to the respective class by reading the sentences.

4.2.1 Problem 1: *false negative matches*

The incomplete KB problem: KBs are usually incomplete because they are manually constructed, and it is not possible to cover all human knowledge or stay current. We took frequent relations, which involve an entity of type PERSON, from Freebase (a large collaborative KB) for analysis. We define the incompleteness $\partial(r)$

of a relation r as follows:

$$\partial(r) = \frac{|\{e\}| - |\{e|\exists e', s.t.r(e,e')\epsilon\mathcal{D}\}|}{|\{e\}|}$$

$\partial(r)$ is the percentage of all persons $\{e\}$ that do not have an attribute e' (with which $r(e,e')$ holds). Table 4.1 shows that 93.8% of persons have no place of birth, and 78.5% of them have no nationality. These are must-have attributes for a person. This shows that Freebase is highly incomplete.

Freebase relation types	Incompleteness
/people/person/education	0.792
/people/person/employment_history	0.923
/people/person/nationality*	0.785
/people/person/parents*	0.988
/people/person/place_of_birth*	0.938
/people/person/places_lived*	0.966

Table 4.1: The incompleteness of Freebase (* are must-have attributes for a person).

We further investigate the rate of false negative matches, as the percentage of entity-pairs that are not listed in Freebase but one of its mentions generated by DS does express a relation in the target set of types. We randomly picked 200 unlabeled bags[3] from each of the two datasets [51] [59] generated by DS, and we manually annotate all relation mentions in these bags. The result is shown in Table 4.2, along with a few examples that indicate a relation holds in the set of false negative matches (bag-level). Both datasets have around 10% false negative matches in the unlabeled set of bags. Taking into consideration that the number of positive bags and unlabeled bags are highly imbalanced (1:134 and 1:37 in the Riedel and KBP dataset respectively, before under-sampling the unlabeled class), the number of false negative matches are 11 and 4 times the number of positive bags in Reidel and KBP dataset, respec-tively. Such a large ratio shows false

[3] 85% and 95.7% of the bags in the Riedel and KBP datasets have only one relation mention.

negatives do have a significant impact on the learning process.

Dataset (training)	# positive bags	# positive : # unlabeled	% are false negatives	# positive : # false negative	has human assessment	Examples of false negative mentions
Riedel	4,700	1:134(BD*)	8.5%	1:11.4	no	(/location/location/contains)... in **Brooklyn** 's **Williamsburg**.
						(/people/person/place_lived) **Cheryl Rogowski** , a farmer from **Orange County**
KBP	183,062	1:37(BD*)	11.5%	1:4	yes	(per:city_of_birth) **Juan Martn Maldacena** (born September 10, 1968) is a theoretical physicist born in **Buenos Aires**
						(per:employee_of)**Dave Matthews**, from the **ABC News**,

Table 4.2: False negative matches on the *Riedel* [51] and *KBP* dataset [59]. All numbers are on bag (pairs of entities) level. *BD** are the numbers before down-sampling the *negative* set to 10% and 5% in *Riedel* and *KBP* dataset, respectively.

4.2.2 Problem 2: *false positive matches*

As shown in Figure 4.1, distant supervision labels ... **Kofi Annan** *warning the* **United Nations** ... as a positive mention for the relation Person:Employee_of because <*Kofi Anna, United Nations*> is listed in the corresponding table in the KB. However, the sentence itself doesnt express the employment relation between *Kofi Annan* and *United Nations.* We call this a *false positive match* for distant supervision.

Generating *false positive matches* is a significant problem for the distant supervision labeling process. Riedel et al. 2010 reported that about 31% of relation mentions labeled are *false positive matches* when aligning Freebase to the New York Times corpus (Sandhaus 2008). Surdeanu et al. 2012 reported 39% *false positive matches* rate at mention-level when aligning Wikipedia infoboxes to the source corpus provided by the KBP shared task.

Instead of applying conventional single-instance learning, previous approaches deal with the label noise by using Multiple Instance Learning ([51]) or Multiple Instance Multiple Label Learning([27, 59]). However, the problem is not completely solved for two reasons:

- Since the knowledge base and the corpus are generated independently, there is no guarantee that *at-least-one* mention expresses the relation.
- Our manual inspectation on the *Riedel* [51] dataset shows that around 50% of the positive bags have only one mention. Therefore the *at-least-one are correct* is a strong and artificial constraint which assumes all of the mentions from these single-instance bags are correct.

4.3 Correcting false negatives: a semi-supervised MIML framework

Our goal is to model the bag-level label noise, caused by the incomplete KB problem, in addition to modeling the instance-level noise using a 3-layer MIL or MIML model (e.g., [59]). We propose a 4-layer model as shown in Figure 4.2.

The input to the model is a list of n bags with a vector of binary labels, either Positive (P), or Unlabled (U) for each relation r. Our model can be viewed as a semi-supervised[4] framework that extends a state-of-the-art Multi-Instance Multi-Label (MIML) model [59]. Since the input to previous MIML models are bags with per-relation binary labels of either P or Negative (N), we add a set of latent variables ℓ which models the true bag-level labels, to bridge the observed bag labels $\mathbf{y}$ and the MIML layers. We consider this as our main contribution of the model. Our hierarchical model is shown in Figure 4.2.

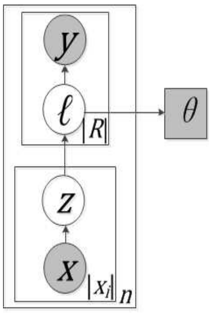


Figure 4.2: Plate diagram of our model.

Let i, j be the index in the bag and mention level, respectively. Following [59], we model mention-level extraction $(p(z_{ij}^r|\mathbf{x}_{ij}; \mathbf{w}_z)$ and multi-instance multi-label aggregation $(p(\ell_i^r|\mathbf{z}_i; \mathbf{w}_\ell^r))$ in the bottom 2 layers. We define:

- r is a relation label. $r \epsilon R \cup \{OTHER\}$, in which OTHER denotes no relation expressed.
- $y_i^r \epsilon \{P, U\}$: r holds for ith bag or it is unlabeled.
- $\ell_i^r \epsilon \{P, N\}$: a hidden variable that denotes whether r holds for the ith bag.
- θ is a observed constant controlling the total number of bags whose latent label is positive.

We define the following conditional probabilities:

[4] We use the term *semi-supervised* because the algorithm uses *unlabeled* bags but existing solutions requires bags to be labeled either *positive* or *negative*.

- $p(y_i^r|\ell_i^r) = \begin{cases} 1/2 & \text{if } y_i^r = P \wedge \ell_i^r = P; \\ 1/2 & \text{if } y_i^r = U \wedge \ell_i^r = P; \\ 1 & \text{if } y_i^r = U \wedge \ell_i^r = N; \\ 0 & \text{otherwise ;} \end{cases}$ It encodes the constraints between true bag-level labels and the entity pair labels in the KB.

- $p(\theta|\ell) \sim \mathcal{N}(\frac{\sum_{i=1}^{n}\sum_{r\epsilon R}\delta(\ell_i^r, P)}{n}, \frac{1}{k})$ where $\delta(x, y) = 1$ if $x = y$, 0 otherwise. k is a large number. θ is the fraction of the bags that are positive. It is an observed parameter that depends on both the source corpus and the KB used.

Similar to [59], we also define the following parameters and conditional probabilities (details are in [59]):

- $z_{ij} \epsilon R \cup \{OTHER\}$: a latent variable that denotes the relation type of the jth mention in the ith bag.

- $\mathbf{x}_{ij}$ is the jth relation mention in the ith bag. We use the set of features in [59].

- $\mathbf{w}_z$ is the weight vector for the multi-class relation mention-level classifier.

- $\mathbf{w}_\ell^r$ is the weight vector for the rth binary top-level classifier (from mention to bag-level

- $p(\ell_i^r|\mathbf{z}_i; \mathbf{w}_\ell^r) \sim Bern(f_\ell(\mathbf{w}_\ell^r, \mathbf{z}_i))$ where f_ℓ is probability produced by the rth top-level classifier, from the mention-label level to the bag-label level.

- $p(z_{ij}^r|\mathbf{x}_{ij}; \mathbf{w}_z) \sim Multi(f_z(\mathbf{w}_z, \mathbf{x}_{ij}))$ where f_z is probability produced by the mention-level classifier[5], from the mentions to the mention-label level.

[5] All classifiers are implemented with L2-regularized logistic regression with Stanford CoreNLP

4.3.1 Training

We use hard Expectation-Maximization (EM) algorithm for training the model. Our objective function is to maximize log-likelihood:

$$L(\mathbf{w}_z, \mathbf{w}_\ell) = logp(\mathbf{y}, \theta | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) = log \sum_\ell p(\mathbf{y}, \theta, \ell | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell)$$

Since solving it exactly involves exploring an exponential assignment space for ℓ, we approximate and iteratively set $\ell^* = \arg\max_\ell p(\ell | \mathbf{y}, \theta, \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell)$

$$\begin{aligned} p(\ell | \mathbf{y}, \theta, \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) &\propto p(\mathbf{y}, \theta, \ell | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) \\ &= p(\mathbf{y}, \theta | \ell, \mathbf{x}) p(\ell | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) \\ &= p(\mathbf{y} | \ell) p(\theta | \ell) p(\ell | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) \end{aligned}$$

Rewriting in log form:

$$\begin{gathered} logp(\ell | \mathbf{y}, \theta, \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) \\ = logp(\mathbf{y} | \ell) + logp(\theta | \ell) + logp(\ell | \mathbf{x}; \mathbf{w}_z, \mathbf{w}_\ell) \\ = \sum_{i=1}^{n} \sum_{r \epsilon R} logp(y_i^r | \ell_i^r) + (-k(\frac{\sum_{i=1}^{n} \sum_{r \epsilon R} \delta(\ell_i^r, P)}{n} - \theta)^2) \\ + \sum_{i=1}^{n} \sum_{r \epsilon R} logp(\ell_i^r | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell) + const \end{gathered}$$

In the E-step, we do a greedy search (steps 5-8 in algorithm 1) in all $p(\ell_i^r | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell)$ and update ℓ_i^r until the second term is maximized. $\mathbf{w}_z$, $\mathbf{w}_\ell$ are the model weights learned from the previous iteration.

After fixed ℓ, we seek to maximize:

$$\begin{aligned} logp(\ell | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell) &= \sum_{i=1}^{n} logp(\ell_i | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell) \\ &= \sum_{i=1}^{n} log \sum_{z_i} p(\ell_i, \mathbf{z}_i | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell) \end{aligned}$$

which can be solved with an approximate solution in [59] (step 9-11): update $\mathbf{z}_i$ independently with: $\mathbf{z}_i^* = \arg\max_{\mathbf{z}_i} p(\mathbf{z}_i | \ell_i, \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell)$. More details can be found in [59].

package

Algorithm 1 Training (**E-step**:2-11; **M-step**:12-15)

```
 1: for $i = 1, 2$ to $T$ do
 2:     $\ell_i^r \leftarrow N$ for all $y_i^r = U$ and $r \epsilon R$
 3:     $\ell_i^r \leftarrow P$ for all $y_i^r = P$ and $r \epsilon R$
 4:     $I = \{< i, r > | \ell_i^r = N\}; I' = \{< i, r > | \ell_i^r = P\}$
 5:     for $k = 0, 1$ to $\theta n - |I'|$ do
 6:         $< i', r' >= \arg\max_{<i,r> \epsilon I} p(\ell_i^r | \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell)$
 7:         $\ell_{i'}^{r'} \leftarrow P; I = I \backslash \{< i', r' >\}$
 8:     end for
 9:     for $i = 1, 2$ to $n$ do
10:         $\mathbf{z}_i^* = \arg\max_{\mathbf{z}_i} p(\mathbf{z}_i | \ell_i, \mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell)$
11:     end for
12:     $\mathbf{w}_z^* = \arg\max_{\mathbf{w}_z} \sum_{i=1}^{n} \sum_{j=1}^{|\mathbf{x}_i|} logp(z_{ij} | \mathbf{x}_{ij}, \mathbf{w}_z)$
13:     for all $r \epsilon R$ do
14:         $\mathbf{w}_\ell^{r(*)} = \arg\max_{w_\ell^r} \sum_{i=1}^{n} p(\ell_i^r | \mathbf{z}_i, \mathbf{w}_\ell^r)$
15:     end for
16: end for
17: return $\mathbf{w_z}, \mathbf{w}_\ell$
```

In the M-step, we retrain both of the mention-level and the aggregation level classifiers.

The full EM algorithm is shown in algorithm 1.

4.3.2 Inference

Inference on a bag $\mathbf{x}_i$ is trivial. For each mention:

$$z_{ij}^* = \arg_{z_{ij} \epsilon R \cup \{OTHER\}} \max p(z_{ij} | x_{ij}, \mathbf{w}_z)$$

Followed by the aggregation (directly with $\mathbf{w}_\ell$):

$$y_i^{r(*)} = \arg_{y_i^r \epsilon \{P,N\}} \max p(y_i^r | \mathbf{z}_i; \mathbf{w}_\ell^r)$$

4.4 Correcting false positives: an extension

We extend the framework in the previous section to deal with the *false positive* problem. In this section we focus on false positives and we assume there are no false negatives in the dataset. For the end-system, we will combine both approaches in this section and the previous section to deal with both problems. Such a combination is straightforward since the two approaches share a common underlining framework

4.4.1 A semi-supervised MIML algorithm

In contrast to previous research that simply trust the bag-level labels assigned by the knowledge base, we instead use the additional layer of latent variables to represent the true bag-level labels. Because there are "*positive*" bags which donot have any positive mentions, we believe this more accurately models the labels generated by DS.

The input to the model is a list of n bags with a vector of binary labels, either *Positive (P)*, or *Negative (N)* for each relation r. Our model can be viewed as a framework that extends the MIML model (Surdeanu et al. 2012), but enable the bag-level labels to take different values from the ones assigned by the KB. Following the previous section, we add a set of latent variables ℓ which models the true bag-level labels, to bridge the observed bag labels $\mathbf{y}$ and the MIML layers. The plate diagram is essentially the same to the one shown in Figure 4.2.

The model is the same to the one described in the previous section except the following:

- $y_i^r \epsilon \{P, N\}$: whether r holds for the ith bag. P denotes $Positive$ and N denotes $Negative$.

- θ is an observed constant (a real number in $[0, 1]$) controlling the total number of bags whose latent label is *positive*. θ is set to be smaller than the percentage of positive bags in all bags generated by DS to deal with *false positives.*

- $p(y_i^r|\ell_i^r) = \begin{cases} 1 & \text{if } y_i^r = P \wedge \ell_i^r = P; \\ 1/2 & \text{if } y_i^r = P \wedge \ell_i^r = N; \\ 1/2 & \text{if } y_i^r = N \wedge \ell_i^r = N; \\ 0 & \text{otherwise ;} \end{cases}$ Similarly, it encodes the constraints between true bag-level labels and the entity pair labels in the KB.

The training is the same, except for 1) we fix $\ell_i^r \leftarrow N$ for all $y_i^r = N$, 2) the **E-step**: instead of searching in the set of *negative* bags, we search in "*positive*" bags to find $\arg\max_{<i,r>} p(\ell_i^r|\mathbf{x}_i; \mathbf{w}_z, \mathbf{w}_\ell)$ and set $\ell_i^r \leftarrow P$ until the first two terms of (1) is maximized. This provides the effects that 1) keep the labels of *negative* bags unchanged, 2) labeled the *positive* bags to correct the incorrectly labeled *positive* labels, assigned by the nave labeling heuristics. We use the same inference procedure to the one described in chapter 4.3.2.

4.5 Implementation details

We implement our model on top of the MIML[59] code base.[6] We use the same mention-level and aggregate-level feature sets as [59]. We adopt the same idea of using cross validation for the E and M steps to avoid overfitting. We initialize our algorithm by sampling 5% unlabeled examples as negative, in essence using 1 epoch of MIML to initialize. Empirically it performs well.

[6] Available at http://nlp.stanford.edu/software/mimlre.shtml

4.6 Experiments

Data set: We use the KBP [29] dataset[7] prepared and publicly released by [59] for our experiment since it is 1) large and realistic, 2) publicly available, 3) most importantly, it is the only dataset that has associated human-labeled ground truth. Any KB held-out evaluation without manual assessment will be significantly affected by KB incompleteness. In KBP dataset, the training bags are generated by mapping Wikipedia infoboxes (after merging similar types following the KBP 2011 task definition) into a large unlabeled corpus (consisting of 1.5 million documents from the KBP source corpus and a complete snapshot of Wikipedia). The KBP shared task provided 200 query named entities with their associated slot values (in total several thousand pairs). We use 40 queries as development dataset (dev), and the rest (160 queries) as evaluation dataset. We set $\theta = 0.25$ by tuning on the dev set and use it in the experi-ments. For a fair comparison, we follow [59] and begin by downsampling the negative class to 5%. We also set T=8 and use the following noisy-or (for ith bag) of mention-level probability to rank predicted types (r) of pairs and plot the precision-recall curves for all experiments.

$$Prob_i(r) = 1 - \prod_j (1 - p(z_{ij} = r|\mathbf{x}_{ij}; \mathbf{w}_z))$$

Correct false negatives: We compare our algorithm (*MIML-semi*) to three algorithms: 1) *MIML* [59], the Multiple-Instance Multiple Label algorithm which labels the bags directly with the KB ($y = \ell$). 2) *MultiR* (denoted as *Hoffmann*) [27], a Multiple-Instance algorithm that supports overlapping relations. It also imposes $y = \ell$. 3) *Mintz++* [59], a variant of the single-instance learning algorithm (chapter 4.2). The first two are stat-of-the-art Multi-Instance Multi-Label algorithms. *Mintz++* is a strong baseline [59] and an improved version of the

[7] Available from Linguistic Data Consortium (LDC). http://projects.ldc.upenn.edu/kbp/data/

original distant-supervision algorithm. [41]. Figure 4.3 shows that our algorithm consistently outperforms all three algorithms at almost all recall levels (with the exception of a very small region in the PR-curve). This demonstrates that by treating unlabeled data set differently and leveraging the missing positive bags, *MIML-semi* is able to learn a more accurate model for extraction. Although the proposed solution is a specific algorithm, we believe the idea of treating unlabeled data differently can be incorporated into any of these algorithms that only use unlabeled data as negative examples.

Correct false positives: Similar to the previous experiment, we evaluate the proposed algorithm (*MIML++*) on the KBP dataset, and compared it to three state-of-the-art algorithms: *MIML* [59], *Hoffmann* [27] and *Mintz++* [59]. Figure 4.4 shows that our algorithm improves precision at most recall levels comparing to all three algorithms. This demonstrates that a more accurate model can be learnt by more appropriately modeling the labeling errors.

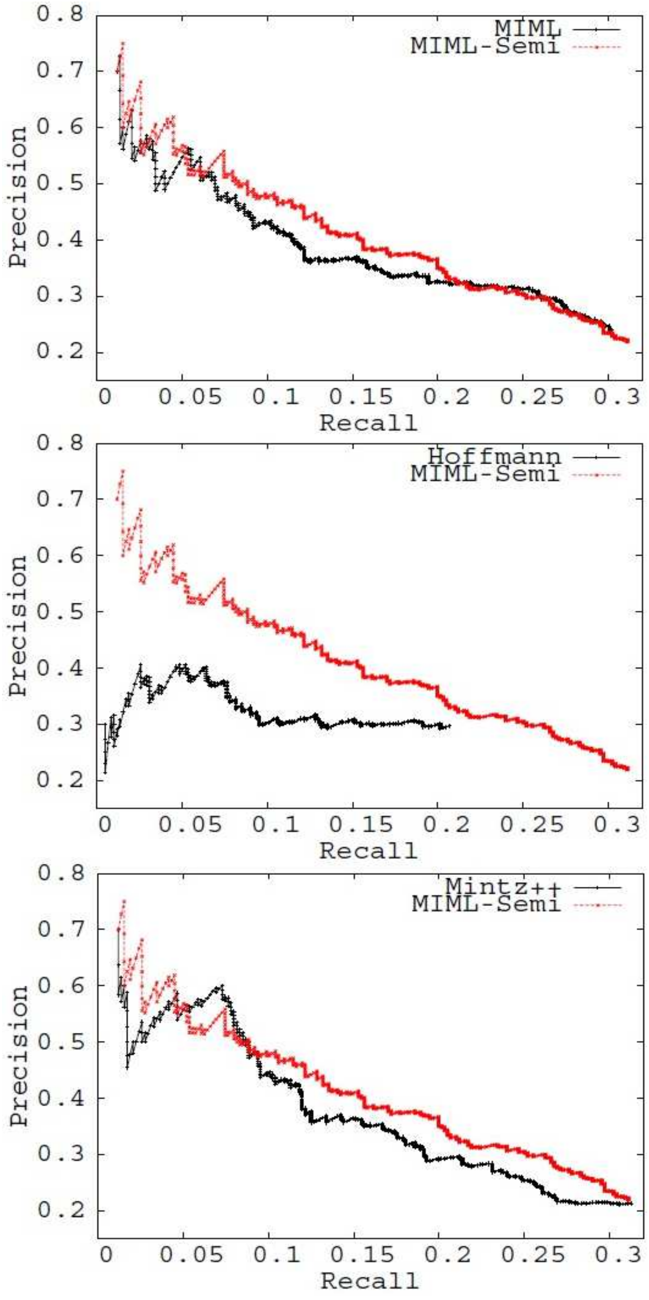


Figure 4.3: Performance on the KBP dataset. The figures on the top, middle and bottom show *MIML*, *Hoffmann*, and *Mintz++* compared to the same *MIML-Semi* curve, respectively. *MIML-Semi* is shown in red curves (lighter curves in black and white) while other algorithms are shown in black curves (darker curves in black and white).

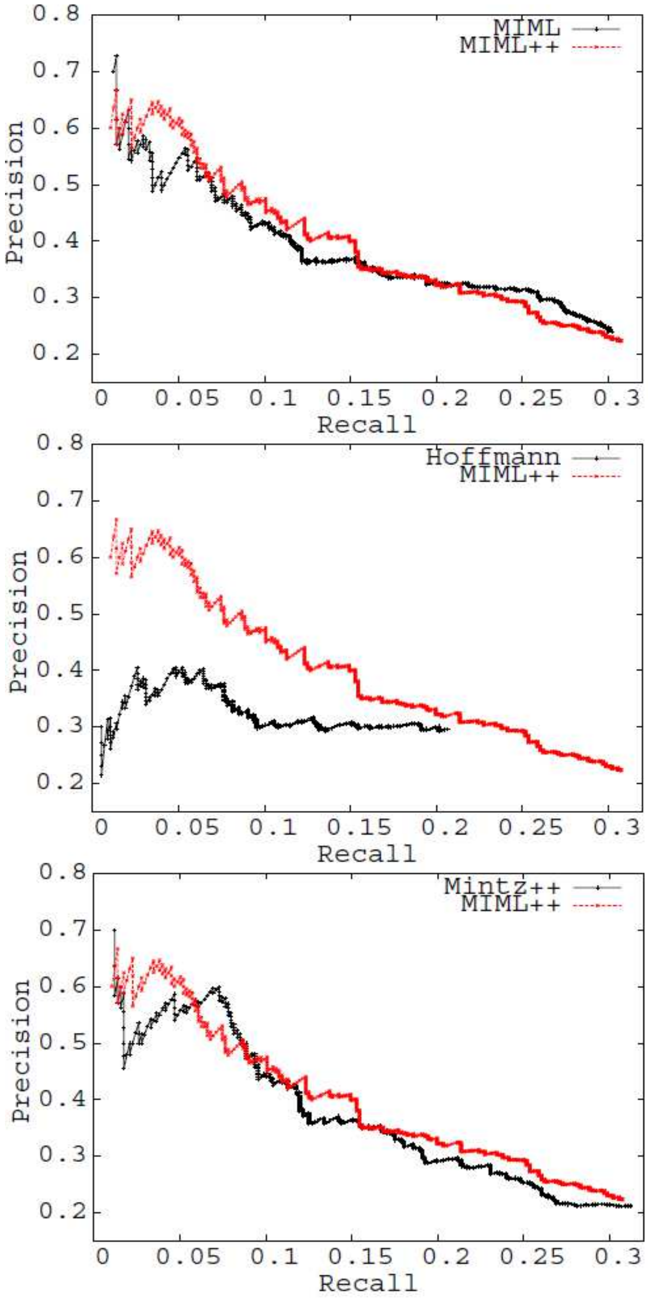


Figure 4.4: Performance on the KBP dataset. The figures on the top, middle and bottom show *MIML*, *Hoffmann*, and *Mintz++* compared to the same *MIML++* curve, respectively. *MIML++* is shown in red curves (lighter curves in black and white) while other algorithms are shown in black curves (darker curves in black and white).

Chapter 5

Unsupervised Relation Extraction via Ensemble Semantics

The Web brings an open-ended set of semantic relations. Discovering the significant types is very challenging. Unsupervised algorithms have been developed to extract relations from a corpus without knowing the relation types in advance, but most of them rely on tagging arguments of predefined types. One recently reported system is able to jointly extract relations and their argument semantic classes, taking a set of relation instances extracted by an open IE (Information Extraction) algorithm as input. However, it cannot handle polysemy of relation phrases and fails to group many similar (synonymous) relation instances because of the sparseness of features. In this chapter, we present a novel unsupervised algorithm that provides a more general treatment of the polysemy and synonymy problems. The algorithm incorporates various knowledge sources which we will show to be very effective for unsupervised relation extraction. Moreover, it explicitly disambiguates polysemous relation phrases and groups synonymous ones.

While maintaining approximately the same precision, the algorithm achieves significant improvement on recall compared to the previous method. It is also very efficient. Experiments on a real-world dataset show that it can handle 14.7 million relation instances and extract a very large set of relations from the Web.

5.1 Introduction

In the era of the Internet, the Web has become a massive potential source of relation mentions. However, there are challenges for Web-scale open-domain relation extraction: the huge and fast-growing scale, a mixed genre of documents and potentially infinite types of relations it carries. To extract these relations, a system should not assume a fixed set of relation types, nor rely on a fixed set of relation argument types. It also should be able to efficiently handle a very large amount of data.

The past decade has seen some promising solutions. Unsupervised relation extraction (URE) algorithms extract relations from a corpus without knowing the relations in advance. However, most algorithms [25] [55] [14] rely on tagging predefined types of entities as relation arguments, and thus are not well-suited for open domain relation extraction.

Recently, [33] proposed Semantic Network Extractor (SNE), which generates argument semantic classes and sets of synonymous relation phrases at the same time. It avoids the requirement of tagging relation arguments of predefined types. However, SNE has 2 limitations: 1) following previous URE algorithms, it only uses features from the set of input relation instances for clustering. Empirically we found that it fails to group many relevant relation instances. These features, such as the surface forms of arguments and lexical sequences in between, are very sparse in practice. In contrast, there exist several well-known corpus-level semantic resources that can be automatically derived from a source corpus and are shown to be useful for generating the key elements of a relation: its 2 argument semantic classes and a set of synonymous phrases. For example, semantic classes can be derived from a source corpus with contextual distributional similarity and web table co-occurrences. The *synonymy*[1] problem for clustering relation instances could potentially be better solved by adding these resources. 2) SNE assumes that each entity or relation phrase belongs to exactly one cluster, thus is not able to effectively handle polysemy of relation phrases[2]. An example of a polysemous phrase is *be the currency of* as in 2 triples <*Euro, be the currency of, Germany*> and <*authorship, be the currency of, science*>. As the target corpus expands from mostly news to the open web, polysemy becomes more important as input covers a wider range of domains. In practice, around 22% (chapter 5.3) of relation phrases are polysemous. Failure to handle these cases significantly limits its effectiveness.

To move towards a more general treatment of the polysemy and synonymy problems, we present a novel algorithm WEBRE for open-domain large-scale unsupervised relation extraction without predefined relation or argument types. The major contributions of this work are:

- WEBRE inorporates a wide range of corpus-level semantic resources for improving relation extraction. The effectiveness of each knowledge source and their combination are studied and compared. To the best of our knowledge, it is the first to combine and compare them for unsupervised relation extrac-

[1] We use the term *synonymy* broadly as defined in chapter 5.3.

[2] A cluster of relation phrases can, however, act as a whole as the phrase cluster for 2 different relations in SNE. However, this only accounts for 4.8% of the polysemous cases

tion.

- WEBRE explicitly disambiguates polysemous relation phrases and groups synonymous phrases, thus it fundamentally avoids the limitation of previous methods.

- Experiments on the Clueweb09 dataset (lemurproject.org/clueweb09.php) show that WEBRE is effective and efficient. We present a large-scale evaluation and show that WEBRE can extract a very large set of high-quality relations. Compared to the closest prior work, WEBRE significantly improves recall while maintaining the same level of precision. WEBRE is efficient. To the best of our knowledge, it handles the largest triple set to date (7-fold larger than largest previous effort). Taking 14.7 million triples as input, a complete run with one CPU core takes about a day.

5.2 Related work

As the preprocessing instance-detection step for the problem studied in this chapter, open IE algorithms extract relation instances (in the form of triples) from the open domain ([18] [2] [20]). For efficiency, they only use shallow features. Reverb ([20]) is a state-of-the-art open domain extractor that targets verb-centric relations, which have been shown in [3] to cover over 70% of open domain relations. [71] filtered relation instances by using a few heuristics and a learning algorithm. Taking the relation instances extracted by open IE algorithms as input, algorithms have been proposed to resolve objects and relation synonyms (Resolver), extract semantic networks (SNE), and map extracted relations into an existing ontology [57].

Recent work shows that it is possible to construct semantic classes automatically with data-driven approaches. They generally fall into three categories. The first category is based on the distributional hypothesis, which states that similar terms tend to appear with similar contexts ([24]), so that it is possible to group similar terms if their contexts are similar. Several previous efforts aimed at utilizing the distributional hypothesis for constructing semantic classes ([47] [43]). The second category ([46] [53]) uses patterns to find similar terms. The third category is language independent approaches ([69] [70]). For example, [69] use HTML wrappers to find similar terms. [49] combine several sources and features for extracting entity classes.

Two tasks are closely related to the task of finding similar phrases for a relation: paraphrase discovery and recognizing textual entailment. Data-driven paraphrase discovery methods ([36] [48] [74] [54]) find paraphrases by extending the idea of distributional similarity to phrases. The Recognizing Textual Entailment algorithms ([4]) can be used for finding related phrases since they find pairs of phrases in which one entails the other.

To efficiently cluster high-dimensional datasets, canopy clustering ([37]) uses a cheap, approximate distance measure to divide data into smaller subsets, and then clusters each subset using an exact distance measure. It has been applied to reference matching. The second phase of WEBRE applies a similar high-level idea of partition-then-cluster for speeding up relation clustering. We design a graph-based partitioning subroutine that uses various types of evidence, such as shared hypernyms. To the best of our knowledge, we have applied the efficient clustering algorithm on the largest set of relation instances extracted from the open domain to date.

5.3 Problem analysis

The basic input is a collection of relation instances (triples) of the form $< ent_1, ctx, ent_2 >$. For each triple, ctx is a relation phrase expressing the relation between the first argument ent_1 and the second argument ent_2. An example triple is <*Obama, win in, NY*>. The triples can be generated by an open IE extractor such as TextRunner or Reverb. Our goal is to automatically build a list of relations, each with the form[3] $\{< ent_1, ctx, ent_2 >\}$ or $< C_1, P, C_2 >$ (P is the set of relation phrases, and C_1 and C_2 are two argument classes). Examples of triples and relations (as Type B relations to be explained in section 5.4.2) are shown in Figure 5.1.

There are two major challenges for building such a list of relations. The first problem is the polysemy of relation phrases, which means that a relation phrase ctx can express different relations in different triples. For example, the meaning of *be the currency of* in the following two triples is quite different: <*Euro, be the currency of, Germany*> and <*authorship, be the currency of, science*>. It is more appropriate to assign these 2 triples to 2 relations *a currency is the currency of a country* and *a factor is important in an area* than to merge them into one. Formally, a relation phrase ctx is polysemous if there exist 2 different relations $< C_1, P, C_2 >$ and $< C_1', P', C_2' >$ where $ctx \epsilon P \bigcap P'$. In the previous example, *be the currency of* is polysemous because it appears in 2 different relations.

Polysemy of relation phrases is not uncommon. We generated clusters from a large sample of triples with the assistance of a soft clustering algorithm, and found that around 22% of relation phrases can be put into at least 2 disjoint clusters

[3] There are 2 possible representations of a relation: as a set of triple instances or a triple with 2 entity classes and a relation phrase class

that represent different relations. More importantly, manual inspection reveals that some common phrases are polysemous. For example, *be part of* can be put into a relation *a city is located in a county* when connecting *Cities* to *Counties*, and another relation *a company is a subsidiary of a parent company* when connecting *Companies* to *Companies*. Failure to handle polysemous relation phrases fundamentally limits the effectiveness of an algorithm. The WEBRE algorithm described later explicitly handles polysemy and synonymy of relation phrases in its first and second phase respectively.

The second problem is the *synonymy* of relation instances. We use the term *synonymy* broadly and we say 2 relation instances are synonymous if they express the same semantic relation between the same pair of semantic classes. For example, both <*Euro, be the currency used in, Germany*> and <*Dinar, be legal tender in, Iraq*> express the relation <*Currencies, be currency of, Countries*>. Solving this problem requires grouping synonymous relation phrases and identifying argument semantic classes for the relation.

Various knowledge sources can be derived from the source corpus for this purpose. In this chapter we pay special attention to incorporating various semantic resources for relation extraction. We will show that these semantic sources can significantly improve the coverage of extracted relations and the best performance is achieved when various resources are combined together.

5.4 Mining relations from the Web

In this section, we first describe the knowledge sources that are used in the relation extraction algorithm, and then introduce the WEBRE algorithm, followed by a brief analysis on its computational complexity.

5.4.1 Knowledge Sources

Entity similarity graph: We build two similarity graphs for entities: a distributional similarity (DS) graph and a pattern-similarity (PS) graph. The DS graph is based on the distributional hypothesis [24], saying that terms sharing similar contexts tend to be similar. We use a text window of size 4 as the context of a term, use Pointwise Mutual Information (PMI) to weight context features, and use Jaccard similarity to measure the similarity of term vectors. The PS graph is generated by adopting both sentence lexical patterns and HTML tag patterns [26] [34] [81]. Two terms (T) tend to be semantically similar if they co-occur in multiple patterns. One example of sentence lexical patterns is *(such as | including) T{,T}* (and|,|.)*. HTML tag patterns include tables, dropdown boxes, etc.

Hypernymy graph: Hypernymy relations are very useful for finding semantically similar term pairs. For example, we observed that a small city in UK and another small city in Germany share common hypernyms such as city, location, and place. Therefore the similarity between the two cities is large according to the hypernymy graph, while their similarity in the DS graph and the PS graph may be very small. Following existing work ([26] [45] [56] [64]), we adopt a list of lexical patterns to extract hypernyms. The patterns include *NP {,} (such as) {NP,}* {and|or} NP*, *NP (is|are|was|were|being) (a|an|the) NP*, etc. In this chapter, we use the terms hypernym and label interchangeably.

Relation phrase similarity: To generate the pairwise similarity graph for relation phrases with regard to the probability of expressing the same relations, we apply a variant of the DIRT algorithm ([36]):

Algorithm 2 Paraphrase Discovery

```
**Input:** A collection of triples $\{< ent_1, ctx, ent_2 >\}$
**Output:** A similarity matrix of phrases $M$
**for all** $t =< ent_1, ctx, ent_2 >$ such that $t \epsilon \{< ent_1, ctx, ent_2 >\}$ **do**
    Collect $< ent_1, ent_2 >$ as features for $ctx$
**end for**
$vec(ctx)$ = feature vector of $ctx$
**for all** $ctx_1 \epsilon \{ctx\}$ **do**
    **for all** $ctx_2 \epsilon \{ctx\}$ **do**
        $sim(ctx_1, ctx_2) = Jaccord(vec(ctx_1), vec(ctx_2))$
        add $sim(ctx_1, ctx_2)$ into $M$
    **end for**
**end for**
**return** $M$
```

Like DIRT, the paraphrase discovery relies on the distributional hypothesis, but there are a few differences: 1) we use stemmed lexical sequences instead of dependency paths as relation phrase candidates. There are two reasons. First, although dependency parsing produces less sparse phrase candidates, it is not applicable to a very large corpus. Second, the impact of the data sparseness problem is reduced in a large corpus. 2) We used ordered pairs of arguments as features of phrases while DIRT uses them as independent features. We empirically tested both feature schemes and found that using ordered pairs results in likely paraphrases but using independent features the result contains general inference rules[4].

5.4.2 WEBRE for Relation Extraction

WEBRE consists of two phases. In the first phase, a set of semantic classes are discovered and used as argument classes for each relation phrase. This results in a

[4] For example, *be part of* has ordered argument pairs $< A, B >$ and $< C, D >$, and *be not part of* has ordered argument pairs $< A, D >$ and $< B, C >$. If arguments are used as independent features, these two phrases shared the same set of features A, B, C, D. However, they are inferential (complement relationship) rather than being similar phrases

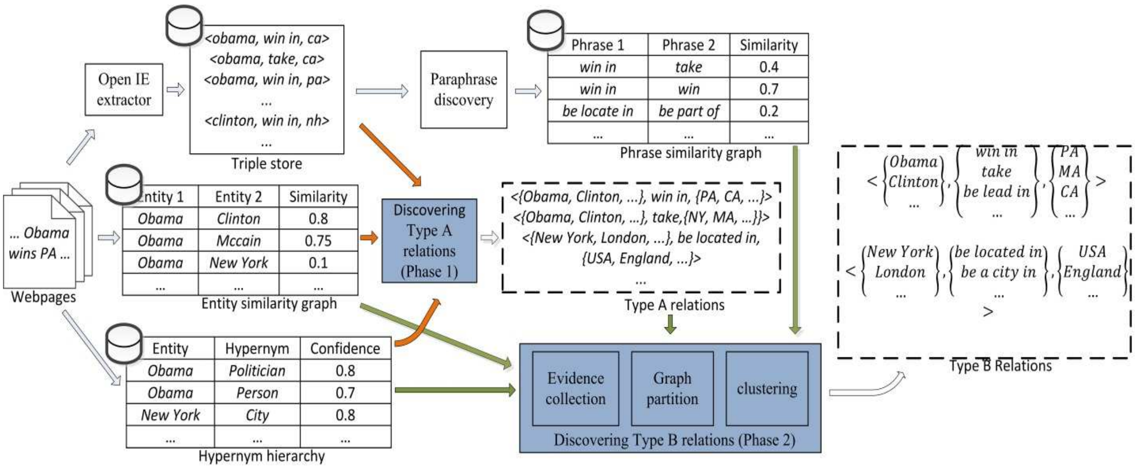


Figure 5.1: Overview of the WEBRE algorithm (Illustrated with examples sampled from experiment results). The tables and rectangles with a database sign show knowledge sources, shaded rectangles show the 2 phases, and the dotted shapes show the system output, a set of Type A relations and a set of Type B relations. The orange arrows denote resources used in phase 1 and the green arrows show the resources used in phase 2.

large collection of relations whose arguments are pairs of semantic classes and which have exactly one relation phrase. We call these relations the Type A relations. An example Type A relation is <{*New York, London*}, *be located in,* {*USA, England,* }>. During this phase, polysemous relation phrases are disambiguated and placed into multiple Type A relations. The second phase is an efficient algorithm which groups similar Type A relations together. This step enriches the argument semantic classes and groups synonymous relation phrases to form relations with multiple expressions, which we called Type B relations. Both Type A and Type B relations are system outputs since both are valuable resources for downstream applications such as Question Answering and Web Search. An overview of the algorithm is shown in Figure 5.1. Here we first briefly describe a clustering subroutine that is used in both phases, and then describe the algorithm in detail.

To handle polysemy of objects (e.g., entities or relations) during the clustering procedure, a key building block is an effective Multi-Membership Clustering algorithm (*MMClustering*). For simplicity and effectiveness, we use a variant of Hierarchical Agglomerative Clustering (HAC), in which we first cluster objects with HAC, and then reassign each object to additional clusters when its similarities with these clusters exceed a certain threshold[5]. The algorithm is the following:

Algorithm 3 MMClustering

```
Input: a vector of objects $I$
       objects similarity function $SimFunc$
       similarity threshold $\alpha$ and $\beta$
Output: clusters of objects $\{C\}$
$\{C\}$ = set each object in $\{I\}$ as a unit cluster
$\{C\} = HAC(\{C\}, \alpha)$
for all $I$ such that $I \epsilon \{I\}$ do
    for all $C$ such that $C \epsilon \{C\}$ do
        if $SimFunc(I, C) > \beta$ then
            Insert $I$ into $C$
        end if
    end for
end for
return $\{C\}$
```

An object can be an entity as in phase 1, or a relation for phase 2. Empirically β should be greater than α to avoid generating duplicated clusters.

Discovering Type A Relations The first phase of the relation extraction algorithm generates Type A relations, which have exactly one relation phrase and two argument entity semantic classes. For each relation phrase, we apply a clustering algorithm on each of its two argument sets to generate argument semantic classes. The Phase 1 algorithm processes relation phrases one by one. For each

[5] This threshold should be slightly greater than the clustering threshold for HAC to avoid generating duplicated clusters

relation phrase *ctx*, step 4 (refer to the Algorithm Phase 1 figure below) clusters the set ent_1 using $MMClustering$ to find left-hand-side argument semantic classes C_1. Then for each cluster C in C_1, it gathers the right-hand-side arguments which appeared in some triples whose left hand-side-side argument are in C, and puts them into ent_2. Following this, it clusters ent_2 to find right-hand-side argument semantic classes. This results in pairs of semantic classes which are arguments of *ctx*. Each relation phrase can appear in multiple Type A relations. For example, <*Cities, be part of, Counties*> and <*Companies, be part of, Companies*> are different Type A relations which share the same relation phrase *be part of.* In the pseudo code, *SimEntFunc* is encoded in the entity similarity graphs.

Algorithm 4 Phase 1: Discovering Type A relations

```
Input: a set of triples $T = \{< ent_1, ctx, ent_2 >\}$
         entity similarity function $SimEntFunc$
         similarity threshold $\alpha$
Output: list of Type A relations $< C_1, ctx, C_2 >$
 1: for each relation phrase $ctx$ do
 2:     $\{< ent_1, ctx, ent_2 >\}$ = set of triples sharing $ctx$
 3:     $\{ent_1\}$ = set of $ent_1$ in $\{< ent_1, ctx, ent_2 >\}$
 4:     $\{C_1\} = MMClustering(\{ent_1\}, SimEntFunc, \alpha)$
 5:     for each $C_1$ in $\{C_1\}$ do
 6:         $ent'_2$=the set of $ent_2$ s.t. $\exists < ent_1, ctx, ent_2 > \epsilon T \bigwedge ent_1 \epsilon C_1$
 7:         $C_2 = MMClustering(\{ent_2\}, SimEntFunc, \alpha)$
 8:         for each $C_2$ in $\{C_2\}$ do
 9:             Add $< C_1, ctx, C_2 >$ into $\{< C_1, ctx, C_2 >\}$
10:         end for
11:     end for
12: end for
13: return $\{< C_1, ctx, C_2 >\}$
```

Discovering Type B Relations The goal of phase 2 is to merge similar Type A relations, such as <*Cities, be located in, Countries*> and <*Cities, be city of, Countries*>, to produce Type B relations, which have a set of synonymous relation phrases and more complete argument entity classes. The challenge for this phase is to cluster a very large set of Type A relations, on which it is infeasible to run a clustering algorithm that does pairwise comparison. Therefore, we designed an evidence-based partition-then-cluster algorithm.

The basic idea is to heuristically partition the large set of Type A relations into small subsets, and run clustering algorithms on each subset. It is based on the observation that most pairs of Type A relations are not similar because of the sparseness in the entity class and the relation semantic space. If there is little or no evidence showing that two Type A relations are similar, they can be put into different partitions. Once partitioned, the clustering algorithm only has to be run on each much smaller subset, thus computation complexity is reduced.

We use 2 types of evidence. They are shared members and shared hypernyms of relation arguments. For example, 2 Type A relations r_1=<*Cities, be city of, Countries*> and r_2=<*Cities, be located in, Countries*> share a pair of arguments <*Tokyo, Japan*>, and a pair of hypernyms <*city, country*>. These pieces of evidence give us hints that they are likely to be similar. As shown in the pseudo code, shared arguments and hypernyms are used as independent evidence to reduce sparseness.

Steps 1 and 2 build an inverted index from evidence to sets of Type A relations. On the graph G whose vertices are Type A relations, steps 3 to 8 set the value of edge weights based on the strength of evidence that shows the end-points are related. The weight of evidence E is calculated as follows:

$$weight(E) = \frac{\#\, shared\, triples\, in\, which\, E\, appears}{max(\#\, classes\, E\, appears)}$$

The idea behind this weighting scheme is similar to that of TF-IDF in that the weight of evidence is higher if it appears more frequently and is less ambiguous

Algorithm 5 Phase 2: Discovering Type B relations

```
Input: A set of Type A relations $\{r\} = \{< C_1, ctx, C_2 >\}$
       Relation similarity function $SimRelnFunc$
       Map from entities to their hypernyms: $M_{entity2label}$
       Similarity threshold $\alpha$
       Edge weight threshold $\mu$
Output: A list of Type B relations $< C_1, P, C_2 >$
1: $\{< ent, \{r'\} >\}$ = build inverted index from argument $ent$ to the set of Type
   A relations $\{r'\}$ on $\{< C_1, ctx, C_2 >\}$
2: $\{< l, \{r'\} >\}$ = build inverted index from hypernym $l$ of arguments to the set
   of Type A relations $\{r'\}$ on $\{< C_1, ctx, C_2 >\}$ with map $M_{entity2label}$
3: for For each $ent$ in $\{< ent, \{r'\} >\}$ do
4:     for For each pair of $r_1$ and $r_2$ s.t. $r_1 \epsilon \{r'\} \bigwedge r_2 \epsilon \{r'\}$ do
5:         $weight_edge(< r_1, r_2 >) + = weight(l)$
6:     end for
7: end for
8: $G(V, E)$ = weighted graph in which $V = \{r\}$
9: for each edge $< r_1, r_2 >$ in $G$ do
10:    if $weight_edge(< r_1, r_2 >) < \mu$ then
11:        Remove edge $< r_1, r_2 >$ from $G$
12:    end if
13: end for
14: $\{CC\} = DFS(G)$
15: for each connected component $CC$ in $\{CC\}$ do
16:    $\{< C_1, ctx, C_2 >\}$ = vertices in $CC$
17:    $\{< C_1', P', C_2' >\} = MMClustering(\{< C_1, ctx, C_2 >\}, SimRelnFunc, \alpha)$
18:    Add $\{< C_1', P', C_2' >\}$ into $\{< C_1, P, C_2 >\}$
19: end for
20: return $\{< C_1, P, C_2 >\}$
```

(appeared in fewer semantic classes during clustering of phase 1). The weighting scheme is applied to both shared arguments and labels.

After collecting evidence, we prune (steps 9 to 11) the edges with a weight less than a threshold μ to remove noise. Then a Depth-First Search (DFS) is called on G to find all Connected Components CC of the graph. These CCs are the partitions of likely-similar Type A relations. We run $MMClustering$ on each CC in CC and generate Type B relations (step 13 to step 16). The similarity of two relations (*SimRelnFunc*) is defined as follows:

$$sim(< C_1, P, C_2 >, < C_1', P', C_2' >) =$$
$$\begin{cases} 0 & \text{if } sim(P, P') < \sigma \ ; \\ min(sim(C_1, C_1'), sim(C_2, C_2')) & \text{else ;} \end{cases}$$

in which $sim(P, P')$ is the average similarity of the 2 sets of relation phrases in the 2 relations, and $sim(C_1, C_1')$ is the average similarity of the 2 sets of argument entities in the 2 relations. These similarities are looked up from the similarity graphs (phrase and entities) constructed with techniques described in chapter 5.4.1.

5.4.3 Computational Complexity

WEBRE is very efficient since both phases decompose the large clustering task into much smaller clustering tasks over partitions. Given n objects for clustering, a hierarchical agglomerative clustering algorithm requires $O(n^2)$ pairwise comparisons. Assuming the clustering task is split into subtasks of size n_1, n_2, ..., n_k, thus the computational complexity is reduced to $O(\sum_1^k {n_i}^2))$. Ideally each subtask has an equal size of n/k, so the computational complexity is reduced to $O(n^2/k)$, a factor of k speed up. In practice, the sizes of partitions are not equal. Taking the partition sizes observed in the experiment with 0.2 million Type A relations as input, the phase 2 algorithm achieves around a 100-fold reduction in pairwise comparisons compared to the agglomerative clustering algorithm. The combination of phase 1 and phase 2 achieves more than a 1000-fold reduction in pairwise comparison, compared to running an agglomerative clustering algorithm directly on 14.7 million triples. This reduction of computational complexity makes the unsupervised extraction of relations on a large dataset a reality. In the experiments with 14.7 million triples as input, phase 1 finished in 22 hours, and the phase 2 algorithm finished in 4 hours with one CPU core.

Furthermore, both phases can be run in parallel in a distributed computing environment because data is partitioned. Therefore it is scalable and efficient for clustering a very large number of relation instances from a large-scale corpus like the Web.

5.5 Experiment

Data preparation We tested WEBRE on resources extracted from the English subset of the Clueweb09 dataset, which contains 503 million webpages. For building knowledge resources, all webpages are cleaned, POS tagged and chunked with in-house tools. We implemented the algorithms described in chapter 5.4.1 to generate the knowledge sources, including a hypernym graph, two entity similarity graphs and a relation phrase similarity graph.

We used Reverb Clueweb09 Extractions 1.1[6] as the triple store (relation instances). It is the complete extraction of Reverb over Clueweb09 after filtering low confidence and low frequency triples. It contains 14.7 million distinct triples with 3.3 million entities and 1.3 million relation phrases. We choose it because 1) it is extracted by a state-of-the-art open IE extractor from the open-domain, and 2) to the best of our knowledge, it contains the largest number of distinct triples extracted from the open-domain and which is publicly available. Reverb triples are in the form of triples $< argument_1$, *relation phrase*, $argument_2 >$ in which the 2 arguments are noun phrases and relation phrases are the lexical sequence

[6] downloaded from *reverb.cs.washington.edu*

in between. An example triple is <*Boston, is located north of, New York*>. To reduce sparseness of the relation phrases, we apply a dictionary-based stemmer to reduce the inflected form of each word to its base form. We also remove stop words (semantically empty words) from the relation phrases.

Evaluation setup The evaluations are organized as follows: we evaluate Type A relation extraction and Type B relation extraction separately, and then we compare WEBRE to its closest prior work SNE. Since both phases are essentially clustering algorithms, we compare the output clusters with human labeled gold standards and report performance measures, following most previous work such as [33] and [25]. Three gold standards are created for evaluating Type A relations, Type B relations and the comparison to SNE, respectively. In the experiments, we set $\alpha = 0.6$, $\mu = 0.1$ and $\sigma = 0.02$ based on trial runs on a small development set of 10k relation instances. We filtered out the Type A relations and Type B relations which only contain 1 or 2 triples since most of these relations are not different from a single relation instance and are not very interesting. Table 5.1 shows the overall statistics of the experiment. 201,246 Type A relations and 84,126 Type B relations are extracted.

Type	Distinct # of instances
Triple	14,728,268
Entity	3,326,830
Relation phrase	1,299,841
Type A relation	201,246
Type B relation	84,126

Table 5.1: Overall statistics of the experiment. The Type A relations are generated with the Label+SIM method and must contain at least 3 triples.

Evaluating Type A relations To understand the effectiveness of knowledge sources, we run Phase 1 multiple times taking entity similarity graphs (matrices) constructed with resources listed below:

- **TS**: Distributional similarity based on the triple store. For each triple $< ent_1, ctx, ent_2 >$, features of ent_1 are ctx and $ctx\, ent_2$; features of ent_2 are ctx and $ent_1\, ctx$. Features are weighted with PMI. Cosine is used as similarity measure.
- **LABEL**: The similarity between two entities is computed according to the percentage of top hypernyms they share.
- **SIM**: The similarity between two entities is the linear combination of their similarity scores in the distributional similarity graph and in the pattern similarity graph.
- **SIM+LABEL** SIM and LABEL are combined. Observing that SIM generates high quality but overly fine-grained semantic classes, we modify the entity clustering procedure to cluster argument entities based on SIM first, and then further clustering the results based on LABEL.

The outputs of these runs are pooled and mixed for labeling. We randomly sampled 60 relation phrases. For each phrase, we select the 5 most frequent Type A relations from each run ($4 \times 5 = 20$[7] Type A relations in all). For each relation phrase, we ask a human labeler to label the mixed pool of Type A relations that share the phrase: 1) The labelers[8] are asked to first determine the major semantic relation of each Type A relation, and then label the triples as good, fair or bad based on whether they express the major relation. 2) The labeler also reads

[7] Here 4 means the 4 methods (the bullet items above) of computing similarity.

[8] 4 human labelers perform the task. A portion of the judgments were independently dual annotated; inter-annotator agreement is 79%. Moreover, each judgment is cross-checked by at least one more annotator, further improving quality.

all Type A relations and manually merges the ones that express the same relation. These 2 steps are repeated for each phrase. After labeling, we create a gold standard *GS1*, which contains roughly 10,000 triples for 60 relation phrases. On average, close to 200 triples are manually labeled and clustered for each phrase. This creates a large data set for evaluation.

We report micro-average of precision, recall and F1 on the 60 relation phrases for each method. Precision (P) and Recall (R) of a given relation phrase is defined as follows. Here R_A and R'_A represents a Type A relation in the algorithm output and *GS1*, respectively. We use t for triples and $s(t)$ to represent the score of the labeled triple t.

$$P = \frac{\sum_{R_A} \sum_{t \epsilon R_A} s(t)}{\sum_{R_A} |R_A|}, R = \frac{\sum_{R_A} \sum_{t \epsilon R_A} s(t)}{\sum_{R'_A} \sum_{t' \epsilon R'_A} s(t')}$$

$s(t)$ is set to 1.0, 0.5 or 0 for t labeled as good, fair and bad, respectively. Examples of labels assigned to triples are listed in Table 5.2.

Label	Score	Example triple ($< Cities, be\, capital\, of, Countries >$)
Good	1.0	$< Baghdad, Iraq >$
Fair	0.5	$< Sukhumi, Abkhazia >$
Bad	0.0	$< Bangalore, Karnataka\ State >$

Table 5.2: Scores for human judgments on triples. The labels are assigned according to whether they belongs to the main relation $< Cities, be\, capital\, of, Countries >$. The second entry is rated 'Fair' because it's not clear that Abkhazia should be classified as a country.

The results are in Table 5.3. Overall, LABEL performs 53% better than TS in F-measure, and SIM+LABEL performs the best, 8% better than LABEL. Applying a simple sign test shows both differences are clearly significant ($p < 0.001$). Surprisingly, SIM, which uses the similarity matrix extracted from full text, has a F1 of 0.277, which is lower than TS. We also tried combining TS and LABEL but did not find encouraging performance compared to SIM+LABEL.

Among the 4 methods, SIM has the highest precision (0.964) when relation phrases for which it fails to generate any Type A relations are excluded, but its recall is low. Manual checking shows that SIM tends to generate overly fine-grained argument classes. If fine-grained argument classes or extremely high-precision Type A relations are preferred, SIM is a good choice. LABEL performs significantly better than TS, which shows that hypernymy information is very useful for finding argument semantic classes. However, it has coverage problems in that the hypernym finding algorithm failed to find any hypernym from the corpus for some entities. Following up, we found that SIM+LABEL has similar precision and the highest recall. This shows that the combination of semantic spaces is very helpful. The significant recall improvement from TS to SIM+LABEL shows that the corpus-based knowledge resources significantly reduce the data sparseness, compared to using features extracted from the triple store only. The result of the phase 1 algorithm with SIM+LABEL is used as input for phase 2.

Algorithm	Precision	Recall	F1
TS	0.842 (0.886)	0.266	0.388
LABEL	0.855 (0.870)	0.481	0.596
SIM	0.755 (**0.964**)	0.178	0.277
SIM+LABEL	0.843 (0.872)	**0.540**	**0.643**

Table 5.3: Phase 1 performance (averaged on multiple runs) of the 4 methods. The highest performance numbers are in bold. (The number in parenthesis is the micro-average when empty-result relation phrases are not considered for the method).

Evaluating Type B relations The goal is 2-fold: 1) to evaluate the phase 2 algorithm. This involves comparing system output to a gold standard constructed by hand, and reporting performance; 2) to evaluate the quality of Type B relations. For this, we will also report triple-level precision.

We construct a gold standard *GS2*[9] for evaluating Type B relations as follows: We randomly sampled 178 Type B relations, which contain 1547 Type A relations and more than 100,000 triples. Since the number of triples is very large, it is infeasible for labelers to manually cluster triples to construct a gold standard. To report precision, we asked the labelers to label each Type A relation (as a whole rather than label each of its triples) contained in this Type B relation as good, fair or bad based on whether it expresses the same relation. For recall evaluation, we need to know how many Type A relations are missing from each Type B relation. We provide the full data set of Type A relations along with three additional resources: 1) a tool which, given a Type A relation, returns a ranked list of similar Type A relations based on the pairwise relation similarity metric in chapter 5.4, 2) DIRT paraphrase collection, 3) WordNet (Fellbaum, 1998) synsets. The labelers are asked to find similar phrases by checking phrases which contain synonyms of the tokens in the query phrase. Given a Type B relation, ideally we expect the labelers to find all missing Type A relations using these resources. We report precision (P) and recall (R) as follows. Here R_B and R'_B represent Type B relations in the algorithm output and *GS2*, respectively. R_A and R'_A represent Type A relations. $s(R_A)$ denotes the score of R_A.

$$P = \frac{\sum_{R_B} \sum_{R_A \epsilon R_B} |R_A| s(R_A)}{\sum_{R_B} \sum_{R_A \epsilon R_B} |R_A|}, R = \frac{\sum_{R_B} \sum_{R_A \epsilon R_B} |R_A| s(R_A)}{\sum_{R'_B} \sum_{R'_A \epsilon R'_B} |R'_A|}$$

$s(R_A)$ is set to 1.0, 0.5 and 0 for good, fair or bad respectively. Examples of labels assigned to type A relations are listed in Table 5.4.

We also ask the labeler to label at most 50 randomly sampled triples from each Type B relation, and calculate triple-level precision as the ratio of the sum

[9] 3 human labelers performed the task. A portion of the judgments were independently dual annotated; inter-annotator agreement is 73%. Similar to labeling Type A relations, each judgment is cross-checked by at least one more annotator, further improving quality.

Label	Score	Example Type A relation	Main relation in Type B
Good	1.0	*<Ben, Aaron,, be younger brother of, Harish, Curt, >*	*Ben, Chris, is brother of Tim, John Wesley,*
Fair	0.5	*<Bill Richardson, Edwards, Gore, should endorse, Hilary Clinton, Obama, Romney>*	*Gore, Edwards, supports Clinton, Obama,*
Bad	0.0	*<Josh, James, , never like, Jamie, Simon, >*	*Ben, Chris, is brother of Tim, John Wesley,*

Table 5.4: Scores for human judgments on Type A relations. The labels are assigned to each Type A relation according to whether they express the main relation in the Type B relation. The main relations in the third columns are the human perceived major relations.

of scores of triples over the number of sampled triples. We use P_{ins} to represent the precision calculated based on labeled triples. Moreover, as we are interested in how many phrases are found by our algorithm, we also include R_{phrase}, which is the recall of synonymous phrases. Results are shown in Table 5.5.

Interval	P	$R(R_{phrase})$	$F1$	P_{ins}	*count*
$[3, 5)$	0.913	0.426 (0.026)	0.581	0.872	52149
$[5, 10)$	0.834	0.514 (0.074)	0.636	0.863	21981
$[10, 20)$	0.854	0.569 (0.066)	0.683	0.883	6277
$[20, 50)$	0.899	0.675 (0.406)	0.771	0.894	2630
$[50, \infty)$	0.922	0.825 (0.594)	0.871	0.929	1089
Overall	0.897	0.684 (0.324)	0.776	0.898	84126

Table 5.5: Performance for Type B relation extraction. The first column shows the range of the maximum sizes of Type A relations in the Type B relation. The last column shows the number of Type B relations that are in this range. The number in parenthesis in the third column is the recall of phrases.

The result shows that WEBRE can extract Type B relations at high precision (both P and P_{ins}). The overall recall is 0.684. Table 5.5 also shows a trend that if the maximum number of Type A relation in the target Type B relation is larger, the recall is better. This shows that the recall of Type B relations depends on the amount of data available for that relation. Some examples of Type B relations extracted are shown in Table 5.9.

Phrase synonymy and polysemy WEBRE disambiguates polysemous relation phrases and groups synonymous ones with the phase 1 and phase 2 algorithms respectively. In Table 5.6, we show the type A relations generated for two relation phrases *withdraw from* and *be a unit of*. We can see that phase 1 effectively placed the phrases into several type A relations with different meanings. For instance, *withdraw from* can represent the relationship between two countries (meaning one country withdraws its forces from the other), a country and an organization (showing that the country is no longer a member of the organization), a player and an event, a team and a sport, etc. Multiple meanings of a relation phrase are successfully identified in the first phase of our algorithm, and multiple type A relations are generated accordingly.

Relation phrases	Type A relations	Argument pairs samples
withdraw from	<*country, country*>	<*America, Vietnam*>; <*Israel, Lebanon*>
	<*country, organization*>	<*Albania, the Warsaw Pact*>; <*Zimbabwe, the Commonwealth*>
	<*force, country*>	<*American forces, Vietnam*>; <*Roman Legions, Britain*>
	<*player, event*>	<*Brandon Bass, the NBA draft*>; <*Agassi, Wimbledon*>
	<*car, sport*>*	<*Chrysler, NASCAR*>; <*Porsche, Grand Prix racing*>
be a unit of	<*company, company*>	<*ABC, the Walt Disney co.*>; <*American Airlines, AMR Corp.*>
	<*unit, concept*>	<*Celsius, temperature*>; <*pounds, weight*>

Table 5.6: Sample relation phrases and their corresponding type A relations. The second column shows the argument class names (assigned label pairs) and the third column shows sample argument pairs. Note: *A wrong pair of labels was assigned by WEBRE based on the hypernym hierarchy, The correct one should be "$< team, sport >$"

We also show the top neighbors (the list of most similar phrase or type A relations, sorted in descending order by their similarities to the query phrase) of a common phrase *be part of* and two Type A relations <*companies, be part of, companies*> and <*cities, be part of, counties*> in Table 5.7. The top neighbors of *be part of* show a mix of two meanings of *be part of*: *a company is a part of its parent company*, or *a city is a part of a county*, whereas the top neighbors of the two type A relations (*be part of* applied to pairs of arguments <*companies, companies*> and <*cities, counties*> respectively) diverge and show a clear split in the meaning of the relations they express. This further shows the importance of applying phase 1 to disambiguate relation phrases.

be part of	<*be part of, company, company*>	<*be part of, city, county*>
be a part of	<*be owned by, company, company*>	<*be located in, city, county*>
be a city in	<*be a division of, company, company*>	<*be the county seat of, city, county*>
be a town in	<*be a unit of, company, company*>	<*be a township in, city, county*>
be a city located in	<*be a subsidiary of, company, company*>	<*be in, city, county*>
be a village in	<*will be acquired by, company, company*>	<*be a village in, city, county*>
be a division of	<*will be purchased by, company, company*>	<*be located in, city, city*>
be a town located in	<*will be bought by, company, company*>	<*be a city located in, city, county*>
be located in	<*be now part of, company, company*>	<*be a city in, city, county*>
be a fact of	<*be a part of, company, company*>	<*be a town located in, town, county*>
be a subsidiary of	<*be recently sold to, company, company*>	<*be a town in, town, county*>
(I)	(II)	(III)

Table 5.7: Top neighbors of a relation phrase and the 2 type A relations it appears in (I: top-10 similar relation phrases of “*be part of*“; II: top-10 similar type A relations of <*companies, be part of, companies*>; III: top-10 similar type A relations of <*cities, be part of, counties*>). To emphasize our focus on the relation phrases, in this table we show the relation phrase first, and then 2 argument classes (assigned “labels“). This is different from previous notation <Argument class 1, relation phrases, Argument class 2>.

Using the large set of Type A and Type B relations extracted by WEBRE, we generate the Cumulative Distribution of Frequency (CDF) of the number of Type A relations the relation phrases belong to, and the CDF of the number of synonymous relation phrases each Type B relation has, and plot them in Figure 5.2 to shows the distribution of polysemy and synonymy of relation phrases empirically.

The top figure in Figure 5.2 shows the CDF of the number of Type A relations to which a relation phrase belongs. Around 40% of phrases can be put into 2 Type A relations, and around 8% of them can be put into at least 5 Type A relations. It shows that WEBRE can disambiguate a large number of relation phrases. In the bottom figure of Figure 5.2, we plot the CDF of the number of synonymous phrases in Type B relations. Close to 30% of Type B relations have at least 2 relation phrases (Note: x-axis of the bottom figure starts from 1 not 0). Some Type B relations have more than 200 relation phrases. Coupled with the phrase level recall shown in Table 5.5, this demonstrates WEBRE's ability to find synonymous relation phrases.

Comparison with SNE We compare Type B relations extracted by WEBRE to the relations extracted by its closest prior work, SNE[10]. We found SNE is not able to handle the 14.7 million triples in a foreseeable amount of time, so we randomly sampled 1 million (1M) triples[11] and test both algorithms on this set. We also filtered out resulting clusters which have only 1 or 2 triples from both system outputs. For comparison purposes, we constructed a gold standard *GS3* as follows: randomly select 30 clusters from both system outputs, and then find similar clusters from the other system output, followed by manually refining the clusters by merging similar ones and splitting non-coherent clusters. *GS3* contains 742 triples and 135 clusters. We report triple-level pairwise precision, recall and F1 for both algorithms against *GS3*, and report results in Table 5.8. We fine-tuned SNE (using grid search, internal cross-validation, and coarse-to-fine parameter tuning), and report its best performance.

Algorithm	Precision	Recall	F1
WEBRE	0.848	0.734	0.787
SNE	0.850	0.080	0.146

Table 5.8: Pairwise precision/recall/F1 of WEBRE and SNE

Table 5.8 shows that WEBRE outperforms SNE significantly in pairwise recall

[10] Obtained from *alchemy.cs.washington.edu/papers/kok08*

[11] We found that SNEs runtime on 1M triples varies from several hours to over a week, depending on the parameters. The best performance is achieved with runtime of approximately 3 days. We also tried SNE with 2M triples, on which many runs take several days and show no sign of convergence. For fairness, the comparison was done on 1M triples.

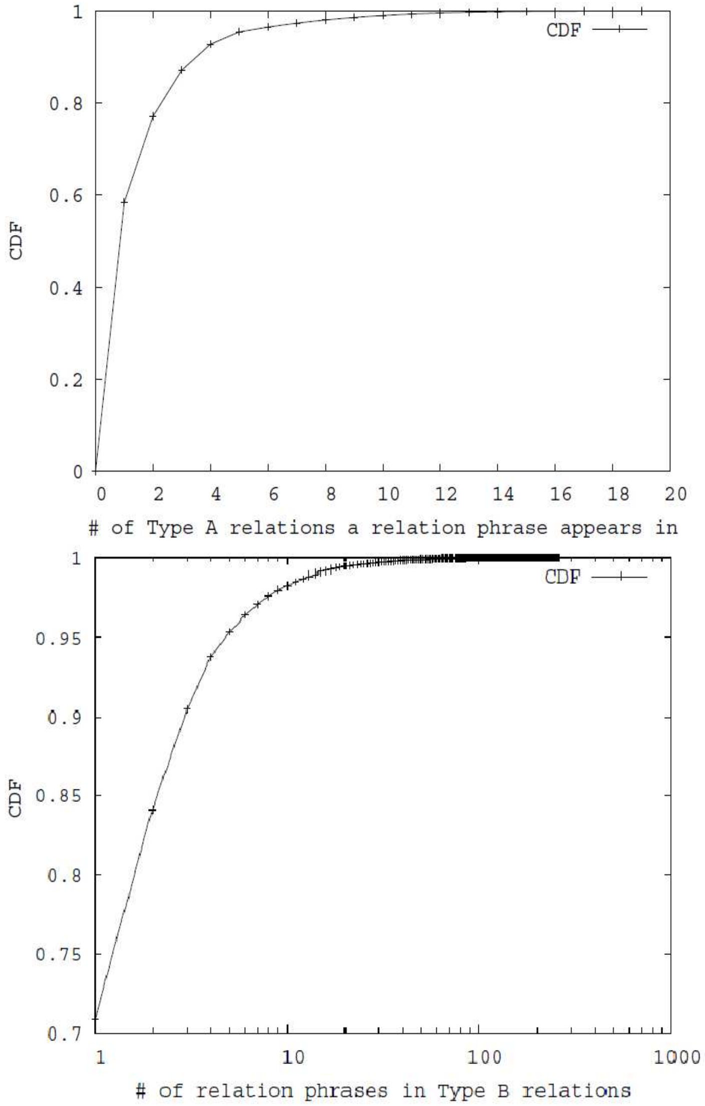


Figure 5.2: Figure on the top shows the Cumulative Distribution of Frequency (CDF) of the number of distinct Type A relations in which a relation phrase appears, and bottom figure shows the CDF of the number of synonymous relation phrases that are in the same Type B relation. Note: x-axis of the bottom figure is in log-scale for presentation.

while having similar precision. There are two reasons. First, WEBRE makes use of several corpus-level semantic sources extracted from the corpus for clustering entities and phrases while SNE uses only features in the triple store. These semantic resources significantly reduced data sparseness. Examination of the output shows that SNE is unable to group many triples from the same generally-recognized fine-grained relations. For example, SNE placed relation instances <*Barbara, grow up in, Santa Fe*> and <*John, be raised mostly in, Santa Barbara*> into 2 different clusters because the arguments and phrases do not share features nor could be grouped by SNEs mutual clustering. In contrast, WEBRE groups them together. Second, SNE assumes a relation phrase to be in exactly one cluster. For example, SNE placed *be part of* in the phrase cluster *be city of* and failed to place it in another cluster *be subsidiary of.* This limits SNEs ability to place relation instances with polysemous phrases into correct relation clusters.

It should be emphasized that we use pairwise precision and recall in Table 5.8 to be consistent with the original SNE paper. Pairwise metrics are much more sensitive than instance-level metrics, and penalize recall exponentially in the worst case[12] if an algorithm incorrectly splits a coherent cluster; therefore the absolute pairwise recall difference should not be interpreted as the same as the instance-level recall reported in previous experiments. On 1 million triples, WEBRE generates 12179 triple clusters with an average size[13] of 13 while SNE generate 53270 clusters with an average size of 5.1. In consequence, pairwise recall drops significantly. Nonetheless, at above 80% pairwise precision, it demonstrates that WEBRE can group more related triples by adding rich semantics harvested from the Web and employing a more general treatment of polysemous relation phrases. On 1M triples,

[12] Pairwise precision and recall are calculated on all pairs that are in the same cluster, thus are very sensitive to cluster sizes. For example, if an algorithm incorrectly split a cluster of size N to a smaller main cluster of size $N/2$ and some constant-size clusters, pairwise recall could drop to as much as $1/4$ of its original value

[13] The clusters which have only 1 or 2 triples are removed and not counted here for both algorithms.

WEBRE finished in 40 minutes, while the run time of SNE varies from 3 hours to a few days.

5.6 Discussion

Domain of the relations The harvested relations are from a wide range of domains. To show the breadth of coverage, we sample a few Type B relations and map them into the relation types defined in a benchmark evaluation, and those defined for two domains, business news and clinical text:

- Automatic Content Extraction (ACE) 2005[14]: ACE is an evaluation on information extraction organized by the U.S. National Institute of Standards and Technology (NIST). It is the standard benchmark for most research in relation extraction. ACE 2005 defines 5 major types and 18 subtypes. They come from a wide range of domains.
- OpenCalais[15] is a service by Thomson Reuters that automatically extracts entities, facts and events from the news domain. A fraction of its facts are similar to relations.
- The 2010 i2b2/VA[16] workshop on Natural Language Processing Challenges for Clinical Records: it defines a few relation types in the clinical record domain.

Table 5.9 presents the Type B relations that can be mapped into relation types defined in ACE 2005, Calais (OpenCalais) or i2b2. Except for artifact subtypes,

[14] http://www.itl.nist.gov/iad/mig/tests/ace/ace05/

[15] http://www.opencalais.com/

[16] https://www.i2b2.org/NLP/Relations

we found WEBRE extraction covered all relation subtypes defined in ACE 2005. This demonstrates that WEBRE's extraction contains lots of relations of interest in a wide range of topics. Furthermore, we show the extracted results contain relations which can be mapped to the representative relations (facts) from Calais and major types in the 2010 i2b2 challenge. In fact, WEBRE extraction covers more than half of the facts (it is sometimes hard to differentiate events from facts in Calais) of Calais and all the three major types in i2b2. This further shows WEBRE extractions wide coverage. We also included two relations that could not be mapped into any types that are defined in the known set.

As a fully unsupervised algorithm, WEBRE extracts a wide range of relations that can be mapped into existing human-defined relation types from a diverse set of domains. This shows it is useful for real-world applications. Furthermore, we show that WEBRE can find relations that have not been defined in previous evaluations (systems). This shows it has the flexibility to discover new open-domain relations.

A hierarchy of relations We evaluate relations (Type B) as a flat set of relations following common practice. However, in practice we observe that some Type A relations gradually merged together when we lowered the threshold in the phase 2 clustering algorithm. In other words, different thresholds can lead to relations at different target granularities. This indicates that relations could form a hierarchy and provides us the ability to query them at different granularity. The following figure illustrates this with sampled results. Building the hierarchy remains an open research topic which we will study in the near future.

Types	Type B relations		
	Argument 1	Relation Phrase	Argument 2
user-owner-inventor-manufacturer	Charlie, Luke, Barbara	drive	a Porsche, a Corvette, an Audi
	AutoCAD, Java, PlayStation	be a trademark of, be a product of	Autodesk, Sun Microsystems, Sony Corporation
	Tim Berners-Lee, Philot. Farnsworth, Thomas Alva Edison	be the inventor of, have invented	the World Wide Web, Television, electricity
	Rockwell Automation, Lexar, HoMedics	be a manufacturer of, be a big name in	programmable controllers, flash memory, massage equipment
citizen-resident-religion-ethnicity	Larry, Grace, Anna	reside in, be a legal resident of	Tulsa, Boulder, Denver
	David, Henry, Charlotte	convert to, be a convert to	Islam, Buddism, Mormonism
org-location	AuctionDrop, BMI, Dr. Pepper	be headquartered in, be a company in	Menlo Park, New York, Plano
employment	Michael Smith, Michael Goldfarb, Matthew Brown	work at, be an employee of	Radioshack, Starbucks, Renaissance magazine
founder Brzezinski, Larson, Bokaer	be founder of, be founding director of	The Trilateral Commission, LINC, Chez Bushwick	
ownership	United Online, Saks Inc., Lockheed Martin	be the parent company to, be the owner of	NetZero, Saks Fifth Avenue, Savi Technology
student-alum	Barnard, Axelrod, Doug	graduate from, be a student at	Simon Fraser University, George Washington University, The University of Maryland
sports-affiliation	Boyd, Mccovey, Tom Brady	sign with, play for	the Boston Red Sox, the San Francisco Giants, the Patriots
investor-shareholder	bond funds, international funds, mutual fund	invest only in, invest primarily in	bonds , foreign stocks, commodities
membership	Elsie, Henry, Boyd	remain a member of, be a member of	Chi Omega Sorority, The Labour Party, Greenpeace
artifact	-	-	-
geographical	Lacey Township, Vernon Township, Woodbridge Township	be a townshiP in be located in	Ocean County, Sussex County, Middlesex County
subsidiary	STM, Stock Building Supply, FedEx Ground	be a subsidiary of, be a part of	SunTrust bank, Wolseley, Federal Express
business	Bohm, Robert, Mike	be a colleague of	Albert Einstein, Werner Heisenberg, Jack Canfield
family	Jean Grey, Jillian, Anne	be a daughter of, be the younger daughter of	John Grey, Jonathan Lee, James
lasting-personal	Paul, Curtis, Kerr	have a close friendship with, developed a friendship with	Michael Jackson, Roosevelt, Bas Rutten
located	David, Henry, Antoine	live in, have resided in	New York, Buffalo, San Francisco
near	Seattle, Samaria, Yorkton	be near, be north of	Portland, Jerusalem, Minot
(Calais) acquisition	Philip Morris, Ingersoll Rand, The Coca-Cola Company	acquire, be buying	Kraft, Trane, Columbia Pictures
(Calais) alliance	Microsoft, Mitsubishi Motors, Enventis	be a strategic partner of	Hewlett-Packard, DaimlerChrysler, Cisco Systems
(Calais) bankruptcy	Lehman Brothers, Penn Central, Syntax-Brillian	go bankrupt in, declare bankruptcy in	September 2008, June 1970, July 2008
(i2b2) medical problems and treatments	Albuterol, Imipramine, Darifenacin	be used for treating, be a medication used to treat	Emphysema, depression, Overactive bladder
(i2b2) test relations with medical problems	Biopsy, blood tests, amnio	come back positive for, confirm the presence of	Breast cancer, Anthrax, Down syndrome
(i2b2) medical problem relations with other medical problems	high blood pressure, Menorrhagia, Chronic Infections	can lead to, may raise the risk of	congestive heart failure, Iron deficiency anemia, weight loss
	C# 2.0, PHP5, Java, C++	allow the use of, also use	destructors, interfaces, template
	Clinton, Obama, McCain,	win in, take	CA, DC, FL, NH, PA, VA, GA, IL

Table 5.9: A list of relations that maps into the ACE 2005 subtypes, representative Calais relation types from news, and i2b2 types from the clinical data domain. The types with (Calais) are representative fact types from Calais. The types with (i2b2) are the major relation types from the 2010 i2b2/VA challenge in clinical text. All other types are relation subtypes defined in ACE 2005. - shows no relations matched. We also show 2 interesting relations that cannot map into the types previously defined. At most 2 relation phrases are shown for each relation.

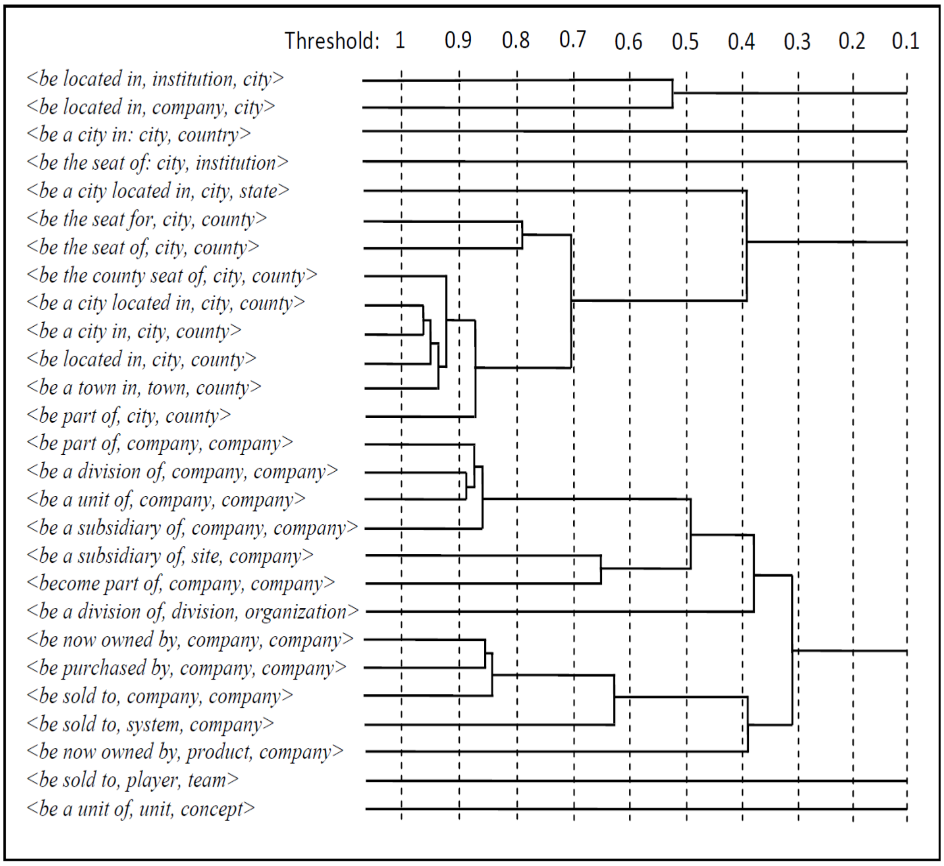

Threshold: 1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1
<be located in, institution, city>
<be located in, company, city>
<be a city in: city, country>
<be the seat of: city, institution>
<be a city located in, city, state>
<be the seat for, city, county>
<be the seat of, city, county>
<be the county seat of, city, county>
<be a city located in, city, county>
<be a city in, city, county>
<be located in, city, county>
<be a town in, town, county>
<be part of, city, county>
<be part of, company, company>
<be a division of, company, company>
<be a unit of, company, company>
<be a subsidiary of, company, company>
<be a subsidiary of, site, company>
<become part of, company, company>
<be a division of, division, organization>
<be now owned by, company, company>
<be purchased by, company, company>
<be sold to, company, company>
<be sold to, system, company>
<be now owned by, product, company>
<be sold to, player, team>
<be a unit of, unit, concept>

Figure 5.3: A hierarchy built from a small sample of type A relations using hierarchical clustering. The top shows the different threshold we used to cluster type A relations. The merging of lines represents the merging of type A relations into the one cluster. To emphasize the meaning of relation phrases applied to a pair of argument classes, in this table we show the relation phrase first, and then 2 argument classes (assigned labels). This is different from previous notation <Argument class1, relation phrases, Argument class2>.

5.7 Conclusion

We have proposed an unsupervised algorithm that can extract relations without predefined types of relations and entities. Compared to previous work, the algorithm handles polysemy of relation instances and achieves a significant improvement in recall while maintaining the same level of precision. We applied the algorithm on a very large web-based dataset and did a large-scale evaluation to show its effectiveness. We analyzed the harvested set of relations in detail and provided some insights into future research on open-domain relation extraction.

Chapter 6

Conclusion and Future Work

In this dissertation, we discuss three techniques for relation extraction. Compared to previous methods for supervised relation extraction, the first method reduces annotation cost by 2/3 while maintaining approximately the same performance. Our second method is a distantly-supervised relation extraction model which provides a better approximation of the weak source of supervision. It improves over the previous approaches. The final method is an ensemble of rich corpus-mined semantic resources for open-domain unsupervised relation extraction. It improves recall significantly over existing methods. In summary, we present a wide range of techniques that perform relation extraction with little or no human supervision, yet outperform state-of-the-art prior methods.

Our immediate future work is in the area of weakly supervised methods for relation extraction. Serious problems remain which limit the broad application of these methods. First, since the labeling heurstic is based on a crude assuption, there are still false positive and false negative matches even after applying our weakly supervised methods. Second, a knowledge base contains many but still a fixed number of relations. A capability to deal with out-of-KB relation types would provide great flexibility to IE systems. Therefore, we need to develop effective algorithms to learn out-of-KB relations.

Our next step includes several folds of research: 1) we will do a comprehensive study of the distant supervision labeling process. In particular, we will align various categories of relations from KBs to various genres of text and analyze the resulting datasets. We will release the analysis and human assessment to the research community. 2) We will re-design the heuristic labeling process to take into account the local context of candidate relation mentions. We will also develop statistical methods to model how likely the KB-assigned relation is expressed by the pairs in the corpus. 3) out-of-KB relation types can be extracted with a bootstrapping procedure which takes a few seeds as input and iteratively finds new pairs and patterns. The key is preventing semantic drift. We will develop a novel bootstrapping algorithm which makes use of background relation topic ([71]), generated with distant supervision, to effectively prevent semantic drift.

Bibliography

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In Proceedings of the fifth ACM conference on Digital libraries, pages 85-94, New York, NY, USA, 2000. ACM.

[2] Michele Banko, Michael J. Cafarella, Stephen Soder-land, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In Proceedings of the International Joint Conference on Artificial Intelligence 2007.

[3] Michele Banko and Oren Etzioni. 2008. The Tradeoffs Between Open and Traditional Relation Extraction. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2008.

[4] Jonathan Berant, Ido Dagan and Jacob Goldberger. 2011. Global Learning of Typed Entailment Rules. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2011.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:9931022, January

[6] Sergey Brin. 1998. Extracting patterns and relations from the world-wide web. In Proceedings of the 1998 International Workshop on the Web and

Databases at the 6th International Conference on Extending Database Technology, EDBT 98, pages 172-183, 1998.

[7] Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In Proceedings of the International Conference on Intelligence Analysis.

[8] Razvan Bunescu and Raymond J. Mooney. 2004. Collective Information Extraction with Relational Markov Networks. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2004.

[9] Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kenrel for relation extraction. In Proceedings of HLT/EMNLP-2005.

[10] Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In Proceedings of NIPS-2005.

[11] Razvan Bunescu and Raymond Mooney. 2007. Learn-ing to extract relations from the web using minimal supervision. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics.

[12] Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology.

[13] Yee Seng Chan and Dan Roth. 2011. Exploiting Syntactico-Semantic Structures for Relation Extraction. In Proceedings of ACL-2011.

[14] Jinxiu Chen, Donghong Ji, Chew Lim Tan, Zhengyu Niu. 2005. Unsupervised Feature Selection for Relation Extraction. In Proceedings of the International Joint Conference on Natural Language Processing 2005.

[15] Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In Proceedings of NIPS-2001.

[16] Thomas G. Dietterich, Richard H. Lathrop, Tomas Lozano-Prez. 1997. Solving the multiple instance problem with axis-parallel rectangles. In Articial Intelligence 89(1-2), 3171 (1997)

[17] Dmitriy Dligach, Rodney D. Nielsen and Martha Palmer. 2010. To annotate more accurately or to annotate more. In Proceedings of Fourth Linguistic Annotation Workshop at ACL 2010

[18] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll (preliminary results). In Proceedings of the International World Wide Web Conference 2004.

[19] Oren Etzioni, Michael Cafarella, Doug Downey, AnaMaria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In Artificial Intelligence, 165(1):91-134.

[20] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2011.

[21] Christiane Fellbaum (Ed.). 1998. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[22] Ralph Grishman. 2012. Information Extraction: Capabilities and Challenges. http://cs.nyu.edu/grishman/tarragona.pdf

[23] Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYUs English ACE 2005 System Description. In Proceedings of ACE 2005 Evaluation Workshop

[24] Zelig S. Harris. 1985. Distributional Structure. The Philosophy of Linguistics. New York: Oxford University Press.

[25] Takaaki Hasegawa, Satoshi Sekine and Ralph Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2004.

[26] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of the International Conference on Computational Linguistics 1992.

[27] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In Proceedings of the Annual Meeting of the Association for Computational Linguistics .

[28] Heng Ji, Ralph Grishman, Hoa Trang Dang and Kira Griffitt. 2010. An Overview of the TAC2010 Knowledge Base Population Track. In Proceedings of TAC-2010

[29] Heng Ji, Ralph Grishman, and Hoa T. Dang. 2011. Overview of the TAC 2011 knowledge base population track. In Proceedings of the Text Analytics Conference.

[30] Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In Proceedings of HLT-NAACL-2007.

[31] Thorsten Joachims. 1999. Transductive Inference for Text Classification using Support Vector Machines. In Proceedings of ICML-1999.

[32] Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In Proceedings of ACL-2004

[33] Stanley Kok and Pedro Domingos. 2008. Extracting Semantic Networks from Text via Relational Clustering. In Proceedings of the European Conference on Machine Learning 2008.

[34] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic Class Learning from the Web with Hypo-nym Pattern Linkage Graphs. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2008.

[35] Xiao-Li Li and Bing Liu. 2003. Learning to classify text using positive and unlabeled data. In Proceedings of IJCAI-2003.

[36] Dekang Lin and Patrick Pantel. 2001. DIRT Discovery of Inference Rules from Text. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2001.

[37] Andrew McCallum, Kamal Nigam, and Lyle Ungar. 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2000.

[38] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text In Proceedings of NAACL-2000.

[39] Bonan Min, Shuming Shi, Ralph Grishman and Chin-Yew Lin. 2012a. Ensemble Semantics for Large-scale Unsupervised Relation Extraction. In Proceedings of EMNLP-CoNLL 2012.

[40] Bonan Min, Xiang Li, Ralph Grishman and Ang Sun. 2012b. New York University 2012 System for KBP Slot Filling. In Proceedings of the Text Analysis Conference (TAC) 2012.

[41] Mike Mintz, Steven Bills, Rion Snow, and Daniel Juraf-sky. 2009. Distant supervision for relation extraction without labeled data. In Proceedings of the 47th An-nual Meeting of the Association for Computational Linguistics.

[42] Truc Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervi-sion from external semantic repositories. In Proceed-ings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.

[43] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Ex-

pansion. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2009.

[44] Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2002.

[45] Patrick Pantel and Deepak Ravichandran. 2004. Automatically Labeling Semantic Classes. In Proceedings of the North American Chapter of the Association for Computational Linguistics Conference 2004.

[46] Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search, In Proceedings of the ACM Conference on Information and Knowledge Management 2004.

[47] Marius Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In Proceedings of the ACM Conference on Information and Knowledge Management 2007.

[48] Marius Pasca and Peter Dienes. 2005. Aligning needles in a haystack: Paraphrase acquisition across the Web. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2005.

[49] Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2009.

[50] Longhua Qian, Guodong Zhou, Qiaoming Zhu and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction . In Proc. of COLING-2008.

[51] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 10).

[52] Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for Unsupervised Relation Identification. In Proceedings of the ACM Conference on Information and Knowledge Management 2007.

[53] Luis Sarmento, Valentin Jijkoun, Maarten de Rijke and Eugenio Oliveira. 2007. "More like these": growing entity classes from seeds. In Proceedings of the ACM Conference on Information and Knowledge Management 2007.

[54] Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In Proceedings of the International Workshop on Paraphrasing, 2005.

[55] Yusuke Shinyama and Satoshi Sekine. 2006. Preemp-tive information extraction using unrestricted relation discovery. In Proceedings of the Human Language Technology Conference of the North American Chapte of the Association for Computation Linguistics.

[56] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning Syntactic Patterns for Automatic Hypernym Discovery. In Proceedings of advances in neural information processing systems 2005.

[57] Stephen Soderland and Bhushan Mandhani. 2007. Moving from Textual Relations to Ontologized Relations. In Proceedings of the 2007 AAAI Spring Symposium on Machine Reading.

[58] Mihai Surdeanu, Sonal Gupta, John Bauer, David Mc-Closky, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2011. Stanfords distant-lysupervised slot-filling system. In Proceedings of the Text Analytics Conference

[59] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In Proceed-ings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning

[60] Ang Sun, Ralph Grishman and Satoshi Sekine. 2011. Semi-supervised Relation Extraction with Large-scale Word Clustering. In Proceedings of ACL-2011.

[61] Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. 2011. New York University 2011 system for KBP slot filling. In Proceedings of the Text Analytics Con-ference.

[62] TAC KBP 2011 task definition. 2011. http://nlp.cs.qc.cuny.edu/kbp/2011/KBP2011_TaskDefinition.pdf

[63] Shingo Takamatsu, Issei Sato, Hiroshi Nakagawa. 2012. Reducing Wrong Labels in Distant Supervision for Relation Extraction. In Proceedings of 50th Annual Meeting of the Association for Computational Lin-guistics.

[64] Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat and Fernando Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2008.

[65] Vladimir N. Vapnik. 1998. Statistical Learning Theory. John Wiley.

[66] David Vickrey, Oscar Kipersztok and Daphne Koller. 2010. An Active Learning Approach to Finding Related Terms. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2010.

[67] Vishnu Vyas and Patrick Pantel. 2009. Semi-Automatic Entity Set Refinement. In Proceedings of the North American Chapter of the Association for Computational Linguistics Conference 2009.

[68] Vishnu Vyas, Patrick Pantel and Eric Crestan. 2009. Helping Editors Choose Better Seed Sets for Entity Set Expansion, In Proceedings of the ACM Conference on Information and Knowledge Management 2009.

[69] Richard C. Wang and William W. Cohen. 2007. Language- Independent Set Expansion of Named Entities Using the Web. In Proceedings of IEEE International Conference on Data Mining 2007.

[70] Richard C. Wang and William W. Cohen. 2009. Automatic Set Instance Extraction using the Web. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2009

[71] Chang Wang, James Fan, Aditya A. Kalyanpur, David Gondek. 2011. Relation Extraction with Relation Topics. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language.

[72] Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In Proceedings of the Interna-tional Conference on Information and Knowledge Management (CIKM-2007).

[73] Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2010.

[74] Hua Wu and Ming Zhou. 2003. Synonymous collocation extraction using translation information. In Proceedings of the ACL Workshop on Multiword Expressions: Integrating Processing 2003.

[75] Limin Yao, Aria Haghighi, Sebastian Riedel, Andrew McCallum. 2011. Structured Relation Discovery Using Generative Models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing 2011.

[76] Alexander Yates and Oren Etzioni. 2007. Unsupervised Resolution of Objects and Relations on the Web. In Proceedings of the North American Chapter of the Association for Computational Linguistics Conference 2007.

[77] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. Journal of Machine Learning Research.

[78] Min Zhang, Jie Zhang and Jian Su. 2006a. Exploring syntactic features for relation extraction using a convolution tree kernel, In Proceedings of HLT-NAACL-2006.

[79] Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006b. A composite kernel to extract relations between entities with both flat and structured features. In Proceedings of COLING-ACL-2006.

[80] Zhu Zhang. 2005. Mining Inter-Entity Semantic Relations Using Improved Transductive Learning. In Proceedings of ICJNLP-2005.

[81] Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing Topic Models for Pattern-based Semantic Class Discovery. In Proceedings of the Annual Meeting of the Association for Computational Linguistics 2009.

[82] Shubin Zhao and Ralph Grishman, 2005. Extracting Relations with Integrated Information Using Kern el Methods. In Proceedings of ACL-2005.

[83] Guodong Zhou, Jian Su, Jie Zhang and Min Zhang. 2005. Exploring various knowledge in relation extraction. In Proceedings of ACL-2005.

[84] Guodong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In Proceedings of EMNLP/CoNLL-2007.